Assessing COVID-19 impact on Dutch SMEs using dynamic network analysis

Master Thesis

Bastijn Kostense

TU Delft

Assessing COVID-19 impact on Dutch SMEs using dynamic network analysis

Master Thesis

by

Bastijn Kostense

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday July 23, 2021 at 13:30.

Student number:	4372972	
Project duration:	November 13, 2020 – July 23, 2021	
Thesis committee:	Dr. H. Wang,	TU Delft, chair, supervisor
	Dr. J. Yang,	TU Delft
	Ir. A. Hovanesyan,	Exact, supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

= exact

Preface

This thesis report concludes the end of my journey of completing my studies at the Delft University of Technology. First, I would like to give a huge thanks to Prof. Dr. Huijuan Wang and Ir. Artur Hovanesyan for their amazing guidance, help, and input throughout this thesis journey. Next, I would like to thank my girlfriend Lise, my dad Nico, my mom Mieke and my sister Amée for their continuous support; I could not have done this without you. My thanks also go out to the whole data science team at Exact; your help, chats, drinks, and the Friday afternoon games were awesome. Lastly, I want to thank all my friends for their support throughout everything; you guys are the best!

<3

Bastijn Kostense
Delft, July 2021

Disclaimer

The information made available by Exact for this research is provided for use of this research only and under strict confidentiality.

Abstract

The COVID-19 pandemic is influencing the Dutch economy heavily. More so, small and medium-sized enterprises, also known as SMEs, are notoriously unstable and as a result, could be even more heavily affected by the coronavirus outbreak. The first major lockdown in The Netherlands was instated on March 23, 2020, which introduced several new measures, such as the prohibition of gatherings, the closing of food and beverage outlets, and the prohibition of all contact-based professions.

In such a time of economic instability as caused by the coronavirus outbreak, it is very useful for a company to know in what financial state they are going to be such that they can actively take precautions, such as liquidating their assets or decreasing their expenses. The financial state of a company is often reflected using Key Performance Indicators, or KPIs for short. These KPIs include metrics like the revenue, cost, and cash flow of a company. The forecasting of these KPIs can help a company in informing in what financial state they are going to be and are usually done using historical data of the company. Whereas the decrease in economic activity of business partners of a company is not reflected in the historical KPI data of the company itself, it can be seen in a network of companies that indicates whether there exists a relationship between two companies by using data on monetary transactions between companies. For this reason, we think that enriching historical KPI data using node features extracted from a dynamic network of companies can help improve the quality of KPI predictions during a period of economic instability such as the COVID-19 pandemic.

This thesis answers the question of whether we can use utilize a dynamic network of SMEs to improve the quality of KPI predictions during the COVID-19 lockdown. To answer this question, we first focus on creating a dynamic network consisting of SMEs and the transactions between them out of unstandardized data by proposing a novel, lightweight entity resolution algorithm that is used to find a mapping between companies. The resulting network is analyzed, and we found that the effects of the coronavirus lockdown are visible in the network. Next, we examine several KPIs, such as the revenue or the cash flow of a company, and we found that we can also see the effects of the COVID-19 lockdown in several KPIs. Lastly, this thesis describes an analysis of whether node features can be used to improve the quality of the forecasting of several of these KPIs, where we found that node features such as the degree and clustering coefficient of a node can indeed help with improving KPI forecasting under certain conditions.

Contents

List of Figures xi

List of Tables xiii

1 Introduction 1
1.1 Problem definition and research questions . . . 2
1.2 Methodologies . . . 2
1.3 Contributions . . . 3
1.4 Report structure . . . 3

2 Related work 5
2.1 Dynamic network construction . . . 5
2.2 Network comparison . . . 5
2.3 Network embeddings . . . 7
2.4 Link Prediction . . . 9
2.4.1 Similarity-based link prediction approaches . . . 9
2.4.2 Probabilistic and maximum likelihood-based link prediction approaches . . . 10
2.4.3 Dimensionality reduction-based link prediction approaches . . . 10
2.4.4 Other link prediction approaches . . . 11
2.5 COVID-19 and SMEs . . . 11

3 SQ1: Creating the dynamic network 13
3.1 Exact data . . . 13
3.2 Entity resolution . . . 13
3.3 Network building . . . 16
3.4 Network analysis . . . 17
3.4.1 Is the size of the network increasing over time? . . . 18
3.4.2 Are closer months in the network more similar to each other than months that are far apart? . . . 19
3.4.3 Can we see the effect of COVID-19 in some metrics in the dynamic network? . . . 22
3.4.4 Are there significant differences in network features between sectors? . . . 24

4 SQ2: Improving KPI predictions using node features 27
4.1 COVID-19 lockdown influence on KPIs . . . 27
4.2 Experimental setup for KPI forecasting . . . 30
4.3 Revenue forecasting . . . 32
4.3.1 Pre-analysis . . . 32
4.3.2 Results . . . 33
4.3.3 Post-analysis . . . 34
4.4 CashFlowMonthly forecasting . . . 36
4.4.1 Pre-analysis . . . 37
4.4.2 Results . . . 37
4.4.3 Post-analysis . . . 39
4.5 D2C_14d forecasting . . . 40
4.5.1 Pre-analysis . . . 41
4.5.2 Results . . . 42
4.5.3 Post-analysis . . . 43

5 Conclusion and future work 45
5.1 Conclusion . . . 45
5.2 Future work . . . 46
5.2.1 Entity resolution . . . 46
5.2.2 KPI forecasting . . . 46
5.2.3 Link prediction . . . 46

A ISIC section codes 47

Bibliography 49

List of Figures

1.1 %-mutation of the GDP in The Netherlands with respect to the previous year per quarter. 1

2.1 Categorization of link prediction approaches as taken from [23]. 10

3.1 Overview of the Exact account data structure. 14
3.2 Overview of the proposed entity resolution algorithm. 14
3.3 Number of fuzzy matching hits per threshold value for Levenshtein similarity. 16
3.4 Degree occurrences in the dynamic network. 17
3.5 Inter-arrival time (in months) occurrences in the dynamic network. 18
3.6 Histogram depicting how many months an edge typically occurs in. 18
3.7 Figures depicting the number of active nodes and edges per month between January 2018 and September 2020. 19
3.8 Figures depicting Jaccard similarity between sets of active nodes. 20
3.9 Figures depicting Jaccard similarity between sets of edges. 20
3.10 Figures depicting the Jensen-Shannon similarity of degree distributions. 21
3.11 Figures depicting the DeltaCon distance. 21
3.12 Figures depicting the NetSimile distance. 22
3.13 Figures depicting the number of active nodes and edges per month, with a lockdown indicator on March 2020. 22
3.14 Number of active nodes per month with the number of type D accounts per month. 23
3.15 Average degree and standard deviation of degree of the subgraph of starting nodes per month. . 23
3.16 Clustering coefficient of the subgraph of starting nodes per month. 24
3.17 Figures verifying whether links are disappearing homogeneously. 25
3.18 Sector occurrences in the dynamic network. 25
3.19 Figures depicting the number of edges between nodes of the same sector and between nodes of different sectors. 25
3.20 Figures depicting the average degree of nodes in different sectors. 26

4.1 Number of nodes for which certain KPIs are calculated. 28
4.2 Figures depicting the average relative change in KPIs for all companies with regard to the same month previous year. 29
4.3 Experimental design of the baseline model and the degree model. 31
4.4 Histogram depicting the Revenue distribution. 32
4.5 Density plots and linear regression lines indicating the correlation between various node features and Revenue. 33
4.6 t-SNE on node2vec embeddings with the Revenue of these nodes. 33
4.7 Percentual change in RMSE and MAE of the degree + clustering model with regard to the baseline model for Revenue prediction per sector. 35
4.8 Histogram depicting the CashFlowMonthly distribution. 37
4.9 Density plots and linear regression lines indicating the correlation between various node features and CashFlowMonthly. 37
4.10 t-SNE on node2vec embeddings with the CashFlowMonthly of these nodes. 38
4.11 Percentual change in RMSE and MAE of the degree model with regard to the baseline model for CashFlowMonthly prediction per sector. 39
4.12 Histogram depicting the D2C_14d distribution. 41
4.13 Density plots and linear regression lines indicating the correlation between various node features and D2C_14d. 41
4.14 t-SNE on node2vec embeddings with the D2C_14d of these nodes. 42

4.15 Percentual change in RMSE and MAE of the degree + clustering model with regard to the baseline model for D2C_14d prediction per sector. 43

List of Tables

2.1 Overview of random-walk based network embedding papers and their application domain. . . 8

3.1 Statistics on the account data after cleaning. 15
3.2 Number of hits for the different matching techniques and the total number of hits. 16

4.1 Results for Revenue prediction. 34
4.2 Results for Revenue prediction for the various degree categories. 36
4.3 Results for Revenue prediction for the various Revenue categories. 36
4.4 Results for Revenue prediction for various months. 36
4.5 Results for CashFlowMonthly prediction. 38
4.6 Results for CashFlowMonthly prediction for the various degree categories. 40
4.7 Results for CashFlowMonthly prediction for the various CashFlowMonthly categories. 40
4.8 Results for CashFlowMonthly prediction for various months. 40
4.9 Results for D2C_14d prediction. 42
4.10 Results for D2C_14d prediction for the various degree categories. 44
4.11 Results for D2C_14d prediction for the various D2C_14d categories. 44
4.12 Results for D2C_14d prediction for various months. 44

1 Introduction

The COVID-19 pandemic is influencing the Dutch economy heavily. More so, small and medium-sized enterprises, also known as SMEs, are notoriously unstable and as a result, could be even more heavily affected by the coronavirus outbreak. As an example of the major influence of the COVID-19 pandemic on the Dutch economy, in figure 1.1[1], the %-mutation of the Gross Domestic Product, or GDP for short, in The Netherlands with respect to the previous year per quarter is shown. In this figure, we can see that overall the GDP is growing fairly steadily every quarter. However, since the coronavirus outbreak at the start of 2020, the GDP has been decreasing, with a major decrease in GDP in the second quarter of 2020.

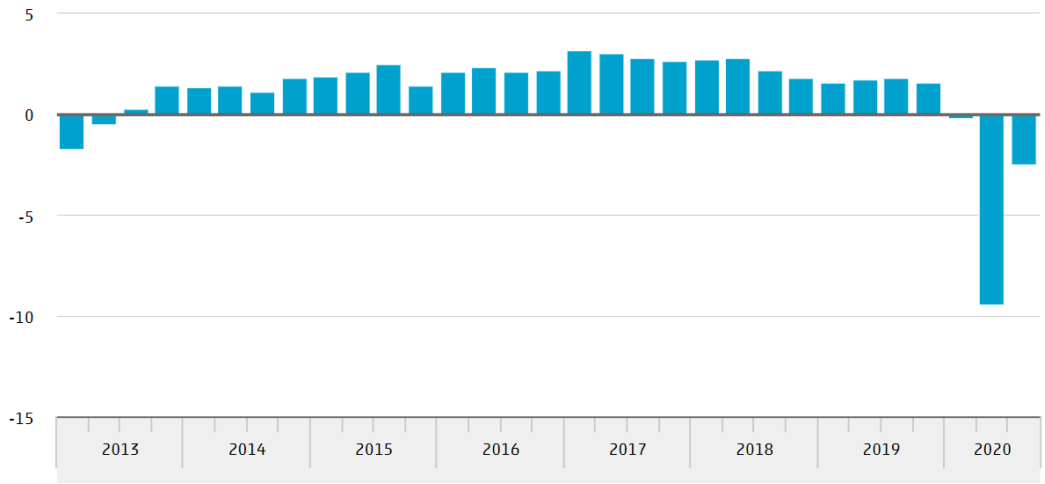


Figure 1.1: %-mutation of the GDP in The Netherlands with respect to the previous year per quarter.

In March 2020, the Dutch government announced the first general measures to limit the spreading of the coronavirus[2]. These general measures included several recommendations such as regularly washing hands, sneezing in the elbow, no more handshaking, and keeping 1.5 meters of distance between each other. The first major lockdown in The Netherlands was instated on March 23, 2020. This lockdown introduced a set of new measures[3], such as advising people to stay inside, the prohibition of all gatherings, the closing of all food and beverage outlets, and the prohibition of all contact-based professions. These measures would last until May 11, 2020, when the first steps towards the relaxation of these measures were taken. Although the COVID-19 pandemic and the resulting restrictions have had a significant impact on the Dutch economy as a whole, the effects of the COVID-19 lockdown on Dutch SMEs remain unknown.

[1] https://www.cbs.nl/nl-nl/nieuws/2020/52/economie-groeit-met-7-8-procent-in-derde-kwartaal-2020
[2] https://www.rijksoverheid.nl/onderwerpen/coronavirus-tijdlijn/maart-2020-maatregelen-tegen-verspreiding-coronavirus
[3] https://www.rijksoverheid.nl/onderwerpen/coronavirus-tijdlijn/nieuws/2020/03/23/aangescherpte-maatregelen-om-het-coronavirus-onder-controle-te-krijgen

In such a time of economic instability as caused by the coronavirus outbreak, it is very useful for a company to know in what financial state they are going to be such that they can actively take precautions, such as liquidating their assets or decreasing their expenses. The financial state of a company is often reflected using Key Performance Indicators, or KPIs for short. These KPIs include metrics like the revenue, cost, and cash flow of a company. The forecasting of these KPIs can help a company in informing in what financial state they are going to be and are usually done using historical data of the company. For example, when predicting the revenue of a company in the upcoming month, the revenues of the company in previous months is used for prediction.

In a time of economic instability, some companies might do less business with each other. For example, when the restaurants during the COVID-19 lockdown were closed, these restaurants did not need to purchase food from their suppliers. Whereas the decrease in economic activity of business partners of a company is not reflected in the historical KPI data of the company itself, it can be seen in a network of companies. For this reason, we think that enriching historical KPI data using node features extracted from a dynamic network of companies can help improve KPI predictions during a period of economic instability such as the COVID-19 pandemic.

Exact is a Dutch company that creates accounting and enterprise resource planning software for SMEs. Over 500 000 SMEs use Exact as their accounting software[4], where the majority of these companies is situated in The Netherlands. The data from Exact describes the monetary transactions among these SMEs and provides the opportunity to explore the impact of the COVID-19 pandemic on individual Dutch SMEs. We are interested in how we can model these transactions as a transaction network consisting of SMEs. We will use this network to see how the positioning of an SME in this network is related to the impact of the COVID-19 lockdown on this SME, and how node features can contribute to the prediction of Key Performance Indicators of these companies.

1.1. Problem definition and research questions

In this thesis, we aim to utilize a dynamic network of SMEs to improve KPI forecasting during times of economic instability. A dynamic (or temporal) network is a network where the links between nodes can vary in whether they are active or inactive per timestep. This thesis will describe the process of preparing unstandardized data on Dutch SMEs from data available at Exact, as well as an analysis on how to improve predictions on KPIs using node features. To work towards the aforementioned goals, we define the following research question:

- *RQ: Can we utilize a dynamic network of SMEs to improve KPI predictions during the COVID-19 lockdown?*

We can split this research question up into two subquestions:

- *SQ1: How can we create a dynamic network of SMEs out of unstandardized data?* To answer this question, we will first develop a novel entity resolution algorithm that is used for network construction. Then, we will construct the dynamic network. Afterward, we will extensively analyze the resulting dynamic network to assess the quality of the network.
- *SQ2: How can we utilize the dynamic network to improve KPI predictions?* To answer this question, we will first check whether we can see the influence of the COVID-19 lockdown on Dutch SMEs by looking at several KPIs and how they have been affected by the coronavirus pandemic. Afterward, we will improve KPI predictions by enriching historical KPI data of a company with node features that we will extract out of the dynamic network as constructed in subquestion 1.

In this thesis, we will try to find answers to these subquestions to be able to ultimately answer our research question.

1.2. Methodologies

To create a dynamic network consisting of companies and transactions between them, an entity resolution algorithm was developed. This entity resolution algorithm uses several data fields, such as the name, Chamber of Commerce number, VAT number, ZIP code, and email address of companies to find matches between

[4] `https://www.exact.com/`

these companies. Then, data on transactions is used to construct the dynamic network of SMEs. After constructing the dynamic network, we will assess the quality of this network by performing an extensive analysis.

Now that we have our resulting dynamic network, to answer the second subquestion, we will take a look at the companies and how they have been influenced by the coronavirus lockdown by looking at the changes in several KPIs over time to assess which of these KPIs have been heavily influenced by the COVID-19 lockdown. We hypothesize that because of the sudden change in the values of these KPIs that is not reflected in the KPI values before as a result of the coronavirus lockdown, enriching the predictions using node features could increase the accuracy of KPI predictions. Therefore, we will improve the forecasting of these KPIs by enriching the historical data with node features. Afterward, we will extensively analyze several scenarios where enriching the historical data with node features is particularly useful for decreasing the error of predictions.

1.3. Contributions

In this work, we develop methods to improve KPI predictions during the COVID-19 lockdown. These KPI predictions are important for SMEs, since knowing the future financial state of the company can help with actively taking precautions, such as liquidating assets or decreasing expenses. Therefore, improving these predictions can be a valuable endeavour. We do this by enriching the prediction data with node features extracted out of a dynamic network of Dutch SMEs. Specifically, the contributions of this thesis can be summarized as follows:

- We propose a novel, lightweight entity resolution algorithm used for network construction out of unstandardized data.
- We analyze the resulting dynamic network to assess the quality of the network.
- We verify whether the impact of the COVID-19 lockdown in The Netherlands can be seen in the KPIs of SMEs and in the node features of these SMEs in the dynamic network.
- We use the dynamic network to improve KPI predictions in a time of economical instability such as the coronavirus pandemic.

1.4. Report structure

In chapter 2, we will take a look at related existing literature. Then, in chapter 3, we will discuss how we can effectively create a dynamic network of SMEs out of unstandardized data using the Exact dataset. In chapter 4, we will assess whether we can see the effect of the coronavirus lockdown in existing KPIs and try to improve upon existing KPI forecasting techniques using node features. Lastly, we will reiterate our findings, answer the research question, and discuss some of the future work that could be done in chapter 5.

2

Related work

In this chapter, we will discuss existing literature on several topics that are related to this thesis. In section 2.1, we will discuss how we can represent a dynamic network and how other works that use the Exact dataset to create a network have approached their problem. Then, in section 2.2, we will take a look at several approaches to network comparison, which is used in section 3.4.2 to assess whether closer months in the dynamic network are more similar than months that are further apart. Since we expect that this is the case, this is done to assess the quality of the network. Network embeddings are discussed in section 2.3, which touches upon our work as described in chapter 4 where we use embeddings to try to improve KPI predictions. In section 2.4, we discuss several approaches to link prediction. These techniques are not used in this thesis, but since we provide an algorithm for creating a dynamic network out of unstandardized data, this could be an interesting piece of future work as discussed in section 5.2. Lastly, in section 2.5, we will discuss the influence of the COVID-19 pandemic on SMEs.

2.1. Dynamic network construction

The main idea for preparing the data is that we create a dynamic network where the nodes are companies that use Exact and the edges represent transactions between those companies as further discussed in chapter 3. A dynamic (or temporal) network is a network where the links between nodes can vary in whether they are active or inactive per timestep. In this section, we will take a look at some literature that touches upon how to represent data as a dynamic network. Also, we will shortly discuss the other works that have modeled the Exact dataset as a network.

The authors of [32] discuss the several ways of representing a dynamic network. One of the main advantages of one of these approaches, namely the one where we represent the dynamic network as a sequence of static networks, is that we can use static network analysis methods on the dynamic network. Since this could prove very useful to us, we, therefore, decided to prepare our data as a dynamic network that consists of individual snapshots that are ordered temporally. The construction of the dynamic network will be further discussed in chapter 3.

This thesis is the first work that covers modeling the Exact data in a dynamic network. However, there exist more works that model the Exact data in a static network. The first of these works is [14], where a static network is created to perform predictions on whether invoices will be paid late. A large theme in [14] is the developed entity resolution algorithm. The need for such an entity resolution algorithm will be discussed in further detail in chapter 3. The second work that builds a static network of the Exact data is [30], where network features are calculated to perform credit scoring predictions for Exact companies.

2.2. Network comparison

Performing network comparisons on slices of the dynamic network is an aspect of this thesis since it is used in section 3.4.2 to assess whether closer months in the dynamic network are more similar than months that are further apart. Since we expect that this is the case, this is done to assess the quality of the network. In this section, we will take a look at various approaches towards network comparison and how to assess network similarity.

[34] gives us a good overview of various approaches in the field of static network comparison. The authors first divide the comparison methods into either Known Node-Correspondence or Unknown Node-correspondence. Network comparison methods from the former category can exploit the knowledge that the two networks that we want to compare have at least a common subset of nodes and tend to come from the same application domain. Since this is the case for this thesis, we will mostly be focusing on Known Node-Correspondence network comparison methods.

The difference between two networks can be measured by the distance between the two networks. This distance metric will ultimately be a trade-off among several features. For this thesis, we will mainly focus on the network comparison techniques that apply to unweighted and undirected networks. One of the simplest approaches that is discussed in [34] is measuring the distance between two adjacency matrices. The adjacency matrix A of a graph G is an n by n matrix, where n is the number of nodes in G. The value of $A_{i,j}$ is 1 if there is an edge between nodes i and j, and 0 otherwise. We can use several distance metrics to calculate the distance between two adjacency matrices, such as the Euclidean distance, the Manhattan distance, the Canberra distance, or the Jaccard distance.

One of the limitations of this approach is that simply measuring the overlap between two edge sets might not be optimal because the importance of all edges is not necessarily the same. This issue is addressed by DeltaCon [16], which assesses the similarity in terms of connectivity between two graphs. DeltaCon first constructs a matrix that keeps track of the pairwise node affinities in both graphs, which is then used to compare differences. A simplified form of Fast Belief Propagation is used to generate the pairwise node affinity matrices S as such:

$$S = [s_{i,j}] = [I + \epsilon^2 D - \epsilon A]^{-1}$$

, where I is the identity matrix, A is the adjacency matrix, D is the matrix with the degree of node i in d_{ii}, and ϵ is a small constant that captures the influence between neighbouring nodes. In this matrix S, the entry s_{ij} denotes the influence node i has on node j in terms of connectivity. The root Euclidean distance is then summed over all elements in the two pairwise node affinity matrices as follows:

$$d = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} (\sqrt{s_{1,i,j}} - \sqrt{s_{2,i,j}})^2}$$

, where d is the DeltaCon distance and s_1 and s_2 are the pairwise node affinity matrices of graph 1 and graph 2 respectively. The complexity of DeltaCon is quadratic with regard to the number of nodes.

A different approach called NetSimile that focuses more on scalability is presented in [5]. Essentially, NetSimile computes a set of features over a network that represents the topology of this network. A signature over this set of features can then be computed and compared with the signature of another network to assess the topological similarity between those two networks.

As discussed above, NetSimile first computes the following set of features for each node i in the network G. This set of features consists of:

1. The degree of i.
2. The clustering coefficient of i.
3. The average number of two-hop away neighbors of i.
4. The average clustering coefficient of the neighbors of i.
5. The number of edges in the egonet of i. The egonet of i consists of the subgraph of G containing only i, the neighbors of i, and the edges between this set of nodes.
6. The number of outgoing edges of the egonet of i.
7. The number of neighbors of the egonet of i.

The writers of [5] found that instead of comparing the feature matrices of two networks, generating signature vectors for these matrices and comparing those results in more efficient comparisons. Therefore, for all features as mentioned above, NetSimile calculates the median, mean, standard deviation, skewness, and kurtosis, and inserts these in the signature vector, resulting in a signature vector of length $7 \cdot 5 = 35$. To compare

these signature vectors, the authors use the Canberra Distance to finally calculate the distance between the two networks. One of the main advantages of this approach is that the computational complexity of NetSimile is linear with regard to the number of edges of the network.

Lastly, we will take a look at [11], where the focus lies on explainability and the low computational cost. The authors propose a similarity metric called Gragnostics, which uses the following set of 10 graph-level features that are all computable in linear time:

1. **Density**. This feature measures the interconnectivity of the vertices in the graph, and is calculated by: $\frac{2\cdot|E|}{|V|\cdot(|V|-1)}$.
2. **Bridge**. A bridge in a network is defined as an edge whose removal will result in a disconnected graph. The value for this feature is calculated by: $\frac{bridge(G)}{|V|-1}$, where $bridge(G)$ indicates the amount of bridges in G.
3. **Disconnection**. This feature indicates how many vertices are disconnected from each other, and it is calculated by: $\frac{|C|-1}{|V|-1}$, where C is the set of all maximally connected components.
4. **Isolation**. This feature measures the fraction of isolated vertices, meaning vertices that have zero edges, and is defined as: $\frac{|\{v\in V:d(v)=0\}|}{|V|}$
5. **Constriction**. This feature measures the number of vertices that are required for any information to be able to reach the whole graph. These vertices are called cut vertices, and their removal will result in a disconnected graph. The constriction feature is calculated by: $\frac{cut(G)}{|V|-2}$, where $cut(G)$ is the number of cut vertices in G.
6. **Line**. This feature measures how similar a graph is to a path graph by measuring the degree of sequential connections. It is the fraction of vertices that have the correct degree as it would have in a path graph, and is calculated as: $\sum_{i=1}^{|V|}\frac{l(i)}{|V|}$, where $l(i)=1$ if $D_i=1$ and $i\leq 2$, or $D_i=2$ and $i>2$, otherwise $l(i)=0$. D is a vector of length $|V|$ and every element in D is the degree of a node, where all elements that are 1 are in the front of D.
7. **Tree**. This feature measures how tree-like a network G is, and it is calculated as: $1-\frac{|E|-(|V|-1)}{|V|\cdot\frac{|V|-1}{2}-(|V|-1)}$.
8. **Star**. This feature measures how much of a graph is like a star, and is calculated as follows: $\sum_{v\in V}\frac{d(v*)-d(v)}{(|V|-1)(|V|-2)}$, where $v*$ indicates the node with the highest degree.
9. **Amount of nodes**. This feature calculates the number of nodes and is defined by: $|V|$.
10. **Amount of links**. This feature calculates the number of links and is defined by: $|E|$.

To compare two networks, these features can be gathered in a vector which we can use to calculate the distance between graphs using various distance metrics.

Lastly, we would like to shortly touch upon graph kernels as discussed in [17]. Graph kernels are often used to perform classification tasks on graphs, but they can also be used to compare graphs. However, one of the main drawbacks of graph kernels is that "due to a pairwise similarity calculation, graph kernel methods suffer significantly from computational bottlenecks" [38], and are therefore not suitable for the Exact dataset due to its large size.

2.3. Network embeddings

Network embeddings are a set of approaches that are used to transform a network into a lower-dimensional vector space and are a widely studied topic. These network embeddings can be used for a variety of tasks. In this thesis, we will use embeddings to try to improve KPI predictions as discussed in chapter 4. The field of embeddings for dynamic networks is still young and being explored. As discussed in section 2.1, one of the main advantages of the way we model our data is that we can also use static graph analysis methods. Therefore, in this section, we will discuss methods that touch upon static network embeddings as well as dynamic network embeddings.

Before we start, we must outline what kind of network embedding we are interested in. In [6], a useful categorization of network embedding methods in terms of graph embedding input and graph embedding

output can be found. As for graph embedding input, the authors first make the distinction between heterogeneous graphs, where nodes can have a class label, and homogeneous graphs. The latter case is applicable for our use case. As for homogeneous graphs, the authors of [6] also note that they can be either weighted or unweighted, and directed or undirected; our graph is both unweighted and undirected, and therefore we will mostly focus our research on embedding methods for homogeneous, undirected, and unweighted graphs. As for graph embedding output, we can choose between embedding the nodes of the network, the edges of the network, the graph as a whole, and a hybrid approach. We will be mainly considering whole-graph embeddings and node embeddings. We will also limit ourselves to random-walk based embedding methods since they are well known for their scalability to large datasets and are therefore applicable to our use case.

In table 2.1, we indicate some of the papers and their application domain with regards to our use case.

	Network type		**Embedding type**	
Paper	**Static network**	**Dynamic network**	**Node embedding**	**Graph embedding**
Perozzi et al. [29]	✓		✓	
Grover et al. [13]	✓		✓	
Tang et al. [33]	✓		✓	
Narayanan et al. [26]	✓			✓
Mahdavi et al. [24]		✓	✓	
Beladev et al. [4]		✓		✓

Table 2.1: Overview of random-walk based network embedding papers and their application domain.

We will first take a look at the static node embeddings. In this case, we can use these methods to embed every static node or graph in our dynamic network that consists of a sequence of static graphs and use these embeddings to compare and analyze nodes or graphs. For static node embedding, we will take a look at two papers that both employ a random-walk based method. The first paper that we discuss is [29], where the authors propose an algorithm called DeepWalk. The authors first state that one of the main advantages is that these random-walk based methods scale very well and can be very easily parallelized, which makes them ideal for our use case, hence they are also used on the Exact data in [14]. The DeepWalk embedding algorithm as discussed in [29] takes a graph G as input and outputs a matrix containing a representation of every node in G. The user also specifies the desired amount of walks that start from each node, as well as the desired walk length. DeepWalk then simulates the random walks by choosing a starting node and randomly follows an edge to a neighbor until the desired walk length is reached. The SkipGram algorithm as presented in [25] is used to update representations of the nodes after every walk. After convergence, the DeepWalk algorithm ultimately outputs a matrix containing vector representations for all nodes in G. Alternatively, [13] proposes node2vec, which is an iteration on DeepWalk. node2vec includes two parameters p and q, allowing the user to switch between bread-first search (BFS) and depth-first search (DFS), which allows the user to emphasize local neighborhoods or global structure. LINE, as proposed in [33], is also an iteration on DeepWalk [29]. The authors state that where DeepWalk is only able to capture first-order proximities, LINE can capture both first-order and second-order proximities. It is also very scalable since it uses a stochastic edge sampling method, also making LINE available for weighted edges.

As opposed to static node embedding, we can also embed the graph as a whole using static graph embedding. The approach to static graph embedding that we will discuss is called graph2vec and is proposed in [26]. graph2vec takes as input a set of graphs for which the embeddings will be learned, and outputs a vector of prespecified length for each graph. First, graph2vec extracts a rooted subgraph around every node in the graph. Then, the representations are updated after each iteration using the SkipGram algorithm as presented in [25]. After convergence, a matrix containing the embedding for every graph in the set of graphs is available. There are a lot of parallels between node2vec [13] and graph2vec [26]; both are based on their respective NLP counterparts (word2vec, as described in [25], and doc2vec, as described in [18], respectively), and both use the SkipGram model with stochastic gradient descent with negative sampling to update the representations.

In contrast to the static embedding methods, the dynamic embedding methods do not have to be run on the individual static graphs or nodes in the dynamic graph. Instead, the dynamic embeddings can be trained on the sequence of graphs or nodes as a whole to capture the evolutional patterns. We will start by discussing several approaches to dynamic node embedding. The first approach that we will discuss is [24]. In this paper, the authors try to modify node2vec as proposed in [13] to be able to generate embeddings for dynamic graphs that capture evolving patterns. To capture this evolving pattern for a certain node, [24] uses the embedding

vector of this node of timestep $t-1$ to generate the embedding for the node in timestep t. dynnode2vec only generates random walks for evolving nodes in a certain timestamp, meaning that only nodes that have just been added to the graph or that experience a change in their edges will receive a new embedding, which can, depending on the graph, result in a very significant speedup.

Alternatively, we can also create an embedding for a dynamic graph as a whole. This is done in [4], where the authors propose a random-walk based method to accomplish this task. The authors mention that their approach can be used to compute temporal graph similarity by comparing snapshots in a sequence of static graphs using the embeddings that are created. Similar to node2vec [13], the authors of [4] make use of parameters p and q to be able to emphasize local neighbourhoods or global structure of the graph. For each node, random walks are generated. Similar to [13] and [12], [4] uses negative sampling to reduce the needed computations for the embeddings. After conversion, a matrix containing the embeddings for each graph in the dynamic graph is outputted. The authors conduct several experiments with real-world datasets and conclude that their approach outperforms many state-of-the-art approaches in terms of graph similarity ranking.

One of the drawbacks of using network embeddings to compare nodes or edges is that explainability is still a problem that is difficult to solve. Although research is being done towards explainable node and graph embeddings, such as in [35], creating explainable embeddings is still an open research field.

2.4. Link Prediction

Since we will be constructing a dynamic graph that is evolving over time, link prediction techniques are related to this thesis. However, as of now, link prediction is out of the scope of this thesis but could be interesting for some future work as further described in section 5.2. [23] makes an important distinction between link prediction in static networks and link prediction in dynamic networks. In the former case, link prediction aims towards finding "missing" links in a static graph, whereas in the latter case, link prediction aims to predict links in a future timestep; for this thesis, we are interested in the latter case. Furthermore, [23] provides a useful categorization of link prediction methods as depicted in figure 2.1. We will use the structure as proposed in this figure to discuss approaches to link prediction. Therefore, we will split up the link prediction approaches into similarity-based link prediction approaches, probabilistic and maximum likelihood-based link prediction approaches, dimensionality reduction-based link prediction approaches, and other link prediction approaches.

2.4.1. Similarity-based link prediction approaches

The first type of approach that [23] defines is similarity-based approaches. Of these approaches, the authors claim that only the local similarity-based methods have a relatively low computational complexity and are easily parallelizable, meaning that they are the only similarity-based approach to link prediction that is potentially suitable for our use case.

These local similarity-based methods calculate indices that consider the direct neighbors of a node. A high value for these indices indicates a high likelihood of the two nodes being connected, meaning that these indices can be used to perform link prediction. The authors of [23] list several of these local similarity-based indices. Some of these that are potentially useful to our use case are:

1. **Common Neighbors Index**. This index was proposed in [27] and indicates the amount of common neighbors of two nodes and is calculated as: $S(x,y) = |\Gamma(x) \cap \Gamma(y)|$, where $\Gamma(x)$ is the set of neighbors of x.

2. **Jaccard Coefficient**. This index normalizes Common Neighbors as such: $S(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$.

3. **Adamic/Adar Index**. This index as proposed in [2] also takes the set of common neighbors into account, but assigns a higher value to neighbors having smaller degrees as such: $\sum_{z \in |\Gamma(x) \cap \Gamma(y)|} \frac{1}{log(k_z)}$, where k_z is the degree of node z.

4. **Resource Allocation Index**. This index as proposed in [39] is very similar to the Adamic/Adar Index, the difference being that it more heavily punishes nodes with a higher degree as such: $\sum_{z \in |\Gamma(x) \cap \Gamma(y)|} \frac{1}{k_z}$.

5. **CAR-based indices**. This set of indices is proposed in [7] is based on the assumption that if the common neighbors of two nodes belong to the same local community, these nodes are more likely to be linked. We can define the **CAR-based Common Neighbor Index** as: $S(x,y) = |\Gamma(x) \cap \Gamma(y)| \cdot \sum_{z \in |\Gamma(x) \cap \Gamma(y)|} \frac{|\gamma(z)|}{2}$, where $\gamma(z)$ is the set of neighbors of z that are also neighbors of x and y. Similarly, we can define the

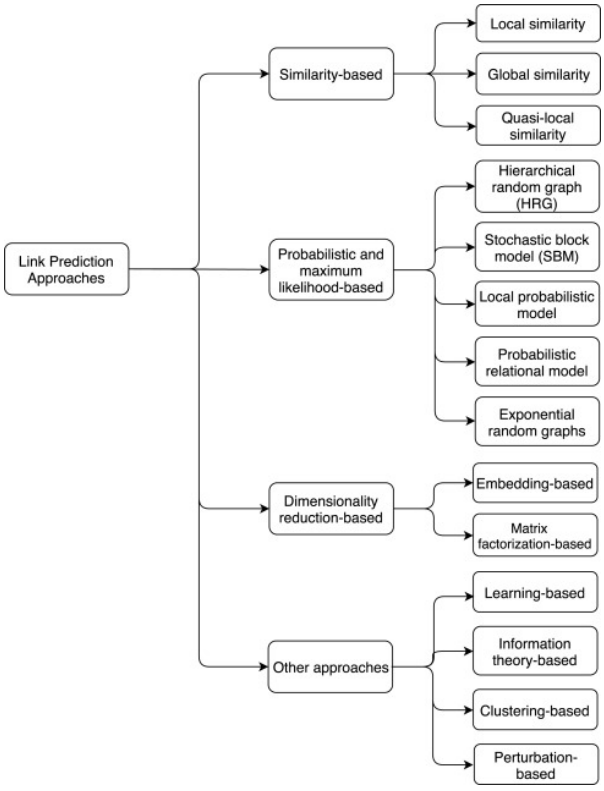


Figure 2.1: Categorization of link prediction approaches as taken from [23].

CAR-based Adamic/Adar Index as: $S(x,y) = \sum_{z \in |\Gamma(x) \cap \Gamma(y)|} \frac{|\gamma(z)|}{log_2(k_z)}$ and the **CAR-based Resource allocation Index** as: $S(x,y) = \sum_{z \in |\Gamma(x) \cap \Gamma(y)|} \frac{|\gamma(z)|}{k_z}$.

6. **Hub Promoted Index**. This similarity index as proposed in [31] is developed especially for networks that consist of a small amount of hub nodes with a high degree, followed by a large amount of nodes with a low degree. The Hub Promoted Index is defined as: $S(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{min(k_x,k_y)}$. Alternatively, the **Hub Depressed Index** [31] is defined as: $S(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{max(k_x,k_y)}$.

2.4.2. Probabilistic and maximum likelihood-based link prediction approaches

As for probabilistic and maximum likelihood-based approaches, [23] states that for probabilistic models, often additional information other than the structural information of the network is needed. Maximum likelihood-based approaches are complex methods with high computational demands, meaning that both probabilistic and maximum likelihood-based approaches are not suitable for our use case.

2.4.3. Dimensionality reduction-based link prediction approaches

Dimensionality reduction-based approaches to link prediction methods can be divided into either embedding-based approaches and matrix factorization-based approaches. The former exploits node embeddings as discussed in section 2.3. These embeddings can be used as input to other machine learning models to generate link predictions. On the other hand, matrix factorization-based approaches extract latent features for each node, resulting in a matrix that contains a representation for each node. These representations can then be used using matrix factorization for link prediction. Both of these approaches are suitable for large networks, making them possibly applicable for this thesis. However, as node embeddings are already discussed in section 2.3, we will not discuss them further in this chapter.

For link prediction in dynamic networks, often Graph Neural Networks, or GNNs for short, are used since they can deal with highly nonlinear structures. These GNNs are essentially similar to the process of first embedding a node into a latent vector and then applying some machine learning model to finally generate a link prediction, the difference being that the entire end-to-end pipeline is provided. [9] provides a deep learn-

ing approach that is tailored towards link prediction in dynamic networks using an encoder-LSTM-decoder architecture. The encoder first encodes the graphs from 1 to $t-1$ to a matrix containing the latent representations for these graphs. The LSTM then learns the evolution patterns from this matrix. Finally, the decoder reconstructs the graph, resulting in the predicted graph for timestep t. This paper claims that they achieve state-of-the-art performance and future work will address reducing the computational complexity of their approach. Alternatively, [19] tries to accomplish the same task of dynamic link prediction, but utilizes an alternative model architecture. First, a Graph Convolutional Network, or GCN for short, is used to capture the local topologies of the network. Then, similarly to [9], an LSTM is used to capture the evolving patterns of the network. Ultimately, a Generative Adversarial Network, or GAN for short, is used to reconstruct the graph. Lastly, we will shortly discuss the model as proposed in [22]. This model encodes the graph using a GCN, similarly to [19]. Attention mechanisms and Gated Recurrent Units, or GRUs for short, are used to extract structures and evolutionary patterns from the inputted graphs. Finally, similarly to [9], a decoder is used to generate a graph which will be the prediction for timestep t.

2.4.4. Other link prediction approaches

The authors of [23] also discuss four other approaches. Learning-based approaches first calculate a set of features over every node in the network. Then, a dataset is created with every combination of nodes where the label is 1 if an edge is present between those nodes and -1 otherwise. Machine learning models can then be applied to this dataset to learn patterns that indicate whether an edge should exist. This model can then be used to perform link prediction. It has to be noted that for some tasks such as graph classification, existing embeddings such as the ones as described in section 2.3 can be used in combination with simple existing machine learning models as discussed in [38]. However, for the task of link prediction, often, these simple machine learning models will not suffice and therefore, GNNs that perform the task in an end-to-end manner will often outperform these solutions. The complexity of this approach depends on the complexity of the features that one chooses. Information theory-based approaches to link prediction are often based on how the topology of a network lends itself to the spreading of information. Clustering-based models try to cluster some specific features together such that links can be predicted. Finally, perturbation-based methods use a novel structural consistency index that indicates how much the structure of a network changes if a certain link is deleted. This index can then be used to perform link predictions.

2.5. COVID-19 and SMEs

To assess the influence of the COVID-19 pandemic on SMEs specifically, several papers exist that touch upon characteristics of both the COVID-19 pandemic and SMEs and how they interact with each other.

[10] discusses the number of bankruptcies of SMEs due to the coronavirus outbreak. The authors suggest that their research indicates that the rate of SMEs going bankrupt could double as a result of the COVID-19 pandemic, which could severely impact the economy as a whole.

The authors of [37] take a look at the potential impact of the COVID-19 pandemic on SMEs and the economy as a whole. One of the main takeaways is that the authors state that the bankruptcy of certain companies could cause a chain reaction which causes other companies to go bankrupt as well. This concept has also been explored in Exact outside the context of the coronavirus outbreak in [30]. Also, [37] states that the COVID-19 pandemic has caused companies to liquidate more of their assets by selling or decreasing the purchase of more risky assets, which could of course affect the long-term prospects of a company.

These long-term prospects on European SMEs are also discussed in [15]. The authors conclude that in the long term many SMEs are forced to upgrade their digital infrastructure, make long-term investments to upgrade production processes, internal as well as external rearrangements will often be necessary. On the flip side, knowledge-based SMEs can apply their technologies to new opportunities that will arise after the COVID-19 pandemic and benefit from subsidies provided by the government. In the short term, most SMEs will have to focus on liquidizing their assets, as also mentioned in [37], as well as fixing issues on the demand and supply side of their supply chains.

3

SQ1: Creating the dynamic network

In this chapter, we will be looking at the first subquestion of how we can create a dynamic network of SMEs out of unstandardized data. As discussed in section 2.1, in previous literature, the authors of [14] and [30] have successfully managed to prepare the data as a static network. However, such a network would fail to capture the temporal aspect of the data. Therefore, we have chosen to prepare our data as a dynamic network. This network will consist of a sequence of static graphs, where each static graph is built with a month of data. In this network, companies that use Exact will be nodes, and an edge between them will be present if there is a transaction between those two companies in the corresponding month. For this thesis, we will consider data from January 2018 to September 2020. In section 3.1, we will discuss the Exact data structure and why an entity resolution algorithm is needed. This entity resolution algorithm will be discussed in section 3.2. In section 3.3, we will discuss the building of the network using the results of the entity resolution algorithm and the data on transactions between companies. Lastly, in section 3.4, we will analyze the network to ensure that the properties and characteristics that we are interested in are captured in this dynamic network.

3.1. Exact data

As discussed before, we will use the data on transactions to create our dynamic network. However, to be able to meaningfully use this transaction data, we must first discuss the account data. In this section, we will discuss how this data is structured.

An overview of the account data structure can be found in figure 3.1. In this image, the green circles represent customers of Exact. These customers are companies that fill out their administration in Exact Online. In Exact Online, the company can enter all kinds of information about itself, such as its name, email address, zip code, website, Chamber of Commerce number, VAT number, and other types of information. These accounts that the Exact customers build for themselves are referred to as Type D accounts. When an Exact customer wants to log a transaction to another company, the customer has to make a profile for that company, where it can input the same types of information as described in the paragraph above. The customer can then log transactions made to this profile. These company profiles are referred to as Type A accounts and are indicated by the red rectangles in figure 3.1. To build the network, we will use the transactions, which are always between type D accounts and type A accounts. However, to be able to successfully create this network, we have to find a matching for the type A account to an existing type D account, as indicated by the dashed arrow line in figure 3.1. The process of finding these matchings is called entity resolution. However, one of the main challenges is that all input fields for both type A and type D accounts are free-form, and all input fields except the name of the company are optional, which poses some challenges. How the entity resolution algorithm works will be discussed in section 3.2.

3.2. Entity resolution

This work is not the first work that performs entity resolution on the Exact account data. As discussed in section 2.1, the author of [14] proposed an entity resolution algorithm. However, since then, the amount of account data has more than doubled. Due to the increase in data, the entity resolution algorithm as proposed in [14] is too complex to be run in a feasible timeframe. Therefore, we propose a more lightweight entity

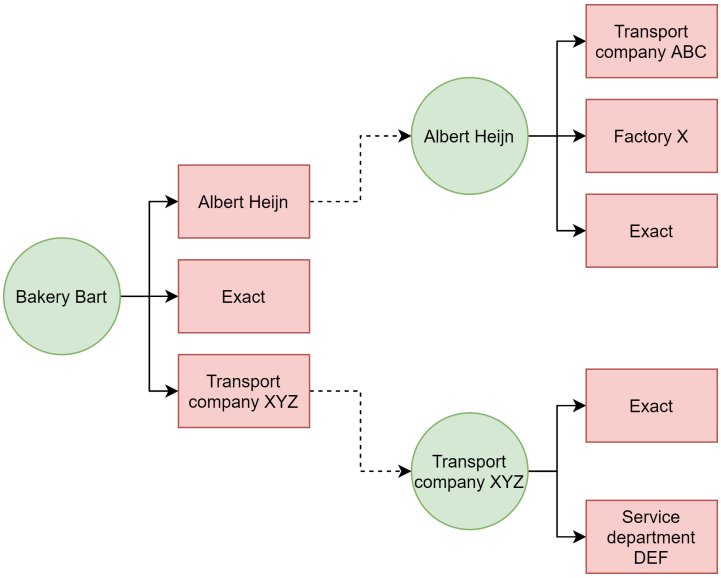


Figure 3.1: Overview of the Exact account data structure.

resolution algorithm that is more scalable than the one proposed in [14]. An overview of the entity resolution algorithm that we will discuss in this section can be found in figure 3.2.

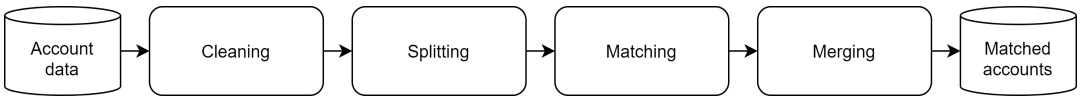


Figure 3.2: Overview of the proposed entity resolution algorithm.

In this algorithm, we try to match type A accounts to type D accounts. We first start with cleaning the data. Afterward, we split the dataset into type D accounts and type A accounts. Then, we try to find a mapping between these two groups. We then merge the results back resulting in the matched accounts. In the following paragraphs, we will go more in-depth into the inner working of the entity resolution algorithm.

The entity resolution algorithm requires the following columns of the account data:

1. **Name**. This is the name of the company, and it is the only column that cannot be empty.
2. **Chamber of Commerce number** or CoC number for short. This is a unique number supplied by the Dutch Chamber of Commerce.
3. **VAT number**. This is a unique string used by governments to track how much tax a business pays.
4. **ZIP Code**. This is the ZIP Code of the company.
5. **Address number**. This is the address number of the company.
6. **Email address**. This is the email address of the company.

When the data enters the algorithm, a cleaning step is performed. As discussed in section 3.1, all input fields are free-form and therefore require cleaning. The following cleaning steps are applied to the columns of the data:

1. **Name**. All names are converted to lowercase, and Dutch business structure indicators such as "bv", "vof", and "eenmanszaak" are removed.
2. **Chamber of Commerce number**. Dutch Chamber of Commerce numbers consist of 8 digits. Alternatively, a Chamber of Commerce number can have 4 additional numbers indicating that it is an establishment of a larger organization. To clean these numbers, we first remove all non-digit characters. If then the data is a number of either length 8 or 12, we include it. Otherwise, we insert the None value.

3. **VAT number**. A Dutch VAT number starts with the two characters "NL", followed by 9 digits, followed by the character "B", followed by 2 digits. To clean these numbers, we first convert all characters to uppercase. Then, if the data matches the description above, we include it. Otherwise, we insert the None value.

4. **ZIP Code**. Dutch ZIP Codes consist of 4 digits followed by 2 letters. To ensure data quality, we first remove spaces and convert the letters to uppercase. If the data then matches the description above, we include it. Otherwise, we insert the None value. This ZIP code is enriched with the **address number** of the company if it is available.

5. **Email address**. We first convert the email address to lowercase. We then check if it starts with at least 1 character, followed by the "@" sign, followed by at least 1 character again, followed by the "." sign, followed by 2 or 3 characters. If this is the case, we include it. Otherwise, we insert the None value.

Statistics on the data after cleaning can be found in table 3.1. In this table, we can see that there are roughly ~800 000 type D and ~234 million type A accounts. In general, type D accounts tend to have more information filled out than type A accounts, especially for the Chamber of Commerce number and VAT number fields.

	Type D accounts	**Type A accounts**
Total number of accounts	780 246	234 335 784
% with name	100	100
% with CoC number	53.4	10.7
% with VAT number	45.1	7.1
% with ZIP Code	69.7	64.1
% with email address	46.9	42.2

Table 3.1: Statistics on the account data after cleaning.

After cleaning, we split the data into type D accounts and type A accounts. This is different than the approach in [14], where all types of accounts are matched. The splitting of data allows for a very large speedup since only a fraction of comparisons is needed.

Afterward, we will start matching the type A accounts to the type D accounts. For this matching we employ three distinct techniques:

1. **Chamber of Commerce number matching**. For this approach, we compare the Chamber of Commerce numbers of the type D and the type A accounts. If they match, we consider the type D account and the type A account to be the same company. However, we cannot have any duplicate Chamber of Commerce numbers in our type D dataset, since two companies with the same Chamber of Commerce numbers cannot exist. Therefore, for the type D accounts only, we exclude all entries with a Chamber of Commerce number that occurs more than 10 times since manual inspection indicates that these Chamber of Commerce numbers are often bogus. For Chamber of Commerce numbers that occur between 2 and 10 times, we drop all duplicates and keep only a single entry.

2. **VAT number matching**. For this approach, we compare the VAT numbers of the type D account and the type A account. If they match, we consider the type D account and the type A account to be the same company. Similar to the Chamber of Commerce number matching, we cannot have VAT numbers in our type D dataset. Therefore, for the type D accounts only, we exclude all entries with a VAT number that occurs more than 10 times. For VAT numbers that occur between 2 and 10 times, we drop all duplicates and keep only a single entry.

3. **Fuzzy matching**. For this matching technique, we compare type D and type A accounts based on their ZIP Code and email address. Although duplicate ZIP codes and email addresses are allowed for type D accounts, we take the same approach as for Chamber of Commerce number matching and VAT number matching, where we exclude all entries with a ZIP code or email address that occurs more than 10 times. For entries with a ZIP code or email that occurs between 2 and 10 times, we drop the duplicates and keep a single entry. We perform this filtering since manual inspection indicates that these ZIP codes and email addresses are often bogus.

If either the ZIP code, which is enriched with the address number, or the email address matches, we calculate the Levenshtein similarity between the names, which is calculated as $1 - l$, where l is the Levenshtein distance [21] between the names of the accounts. We then filter out hits where the Levenshtein distance between the names is too large. To gain some insight into how the decision boundary of the Levenshtein distance impacts the number of fuzzy matching hits, we plot the number of hits per decision boundary in figure 3.3.

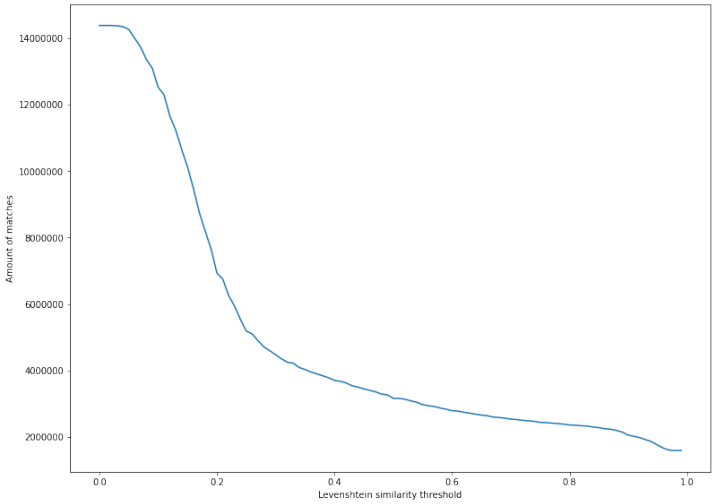


Figure 3.3: Number of fuzzy matching hits per threshold value for Levenshtein similarity.

In figure 3.3, we see that with a low decision boundary, the number of fuzzy matching hits quickly drops. However, when the needed Levenshtein similarity for two records to be considered a hit increases, the number of fuzzy matching hits does not drop as drastically. After inspecting this figure and manually inspecting the matches to see whether they are of high quality, we have decided to utilize a Levenshtein decision boundary of 0.95.

After the matching process, we can merge the results of all three matching processes. The number of hits as a result of the different matching techniques and the total number of distinct hits can be found in table 3.2. In this table, we can see that Chamber of Commerce number matching yields the most hits between type D and type A accounts with ~4.7 million hits. VAT number matching yields ~2.9 million hits, whereas the fuzzy matching technique yields ~1.8 million hits. We can also see that CoC number matching, VAT number matching, and fuzzy matching yield a total of ~9.3 million hits, but we only have a total of ~6.9 million distinct hits, meaning that ~2.4 million hits were duplicates.

	Number of hits
CoC number matching	4 717 751
VAT number matching	2 857 814
Fuzzy matching	1 767 134
Total	6 920 017

Table 3.2: Number of hits for the different matching techniques and the total number of hits.

3.3. Network building

To build the dynamic network, we are going to use data on transactions. To recap, a transaction always happens between a type D account and a type A account. To build our dynamic network, we will only consider transactions where the type A account of the transaction is matched to a type D account using the entity resolution algorithm as described in section 3.2. As discussed before, transactions from January 1st, 2018 to September 30th, 2020 will be considered. We have chosen our starting point to be January 1st, 2018 because we believe that this is an optimum between not having too much data to process but still being able to capture annually occurring temporal patterns. We have chosen September 30th, 2020 as our ending date since

Exact users can log their transactions whenever they want, meaning that the data is often not up-to-date. As a rule of thumb, the Exact data scientists often assume the data on transactions to be complete after three to four months. This meant that at the time of building the network, data up to this point was considered to be complete. Our dynamic network will consist of a series of static networks, one for each month in the period between January 2018 and September 2020, where an edge is present between two nodes if the respective companies had at least one transaction between them in the corresponding month. Additionally, the nodes in the network are enriched with data on in which sector, indicated by the ISIC section code of the company as elaborated on in appendix A, they operate. Also, several Key Performance Indicators, or KPIs for short, are included in the network for the companies they are available for. These KPIs will be discussed more extensively in chapter 4.

3.4. Network analysis

In this section, we will try to get more insight into the network and how it is structured. We will start by looking at the distribution of degrees by iterating over all nodes over all months and retrieving the degrees. The results can be found in figure 3.4. As we can see in this histogram, the degree distribution follows a power-law distribution.

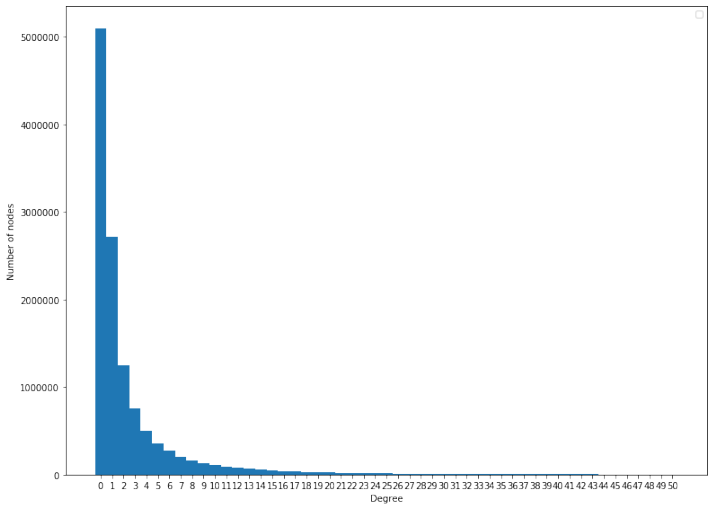


Figure 3.4: Degree occurrences in the dynamic network.

Additionally, we can take a look at the inter-arrival times in our dynamic network. If there are n connections between two nodes, we can calculate $n-1$ inter-arrival times by calculating the number of months between consecutive connections. We can summarize all these inter-arrival times in a histogram as depicted in figure 3.5. Similar to the degree distribution, the inter-arrival time distribution also seems to follow a power-law distribution.

Another informative statistic can be gathered by iterating over all edges and checking in how many months they occur. This gives us an idea of how persistent our edges are through multiple months. A histogram containing this data is depicted in figure 3.6. In this figure, we can see that the most occurring number of months and edge occurs in is 1. However, we can also see that there seems to be a nice distribution where we have an ample amount of data over various numbers of months that an edge occurs in.

Additionally, we will define a set of questions that will help us in verifying the usefulness of this network. Since we want to assure the quality of the resulting dynamic network, we will verify whether the answers to the questions we define adhere to our expectations. In this section, we will analyze the network to get an answer to the following questions:

1. Is the size of the network increasing over time?
2. Are closer months in the network more similar to each other than months that are far apart?
3. Can we see the effect of COVID-19 in some metrics in the dynamic network?
4. Are there significant differences in network features between sectors?

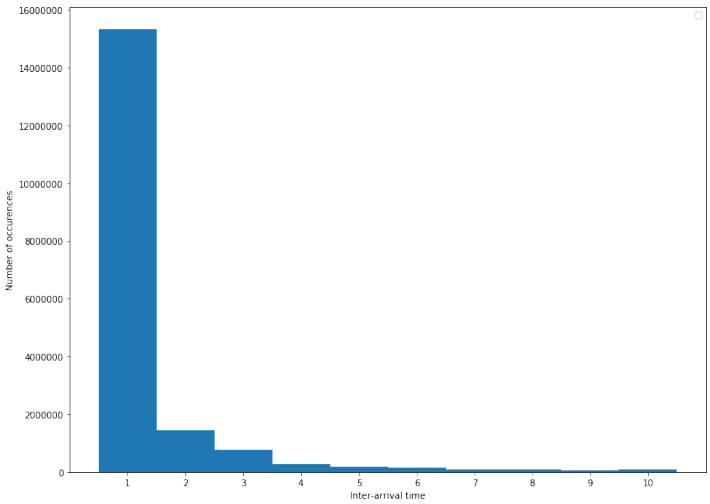


Figure 3.5: Inter-arrival time (in months) occurrences in the dynamic network.

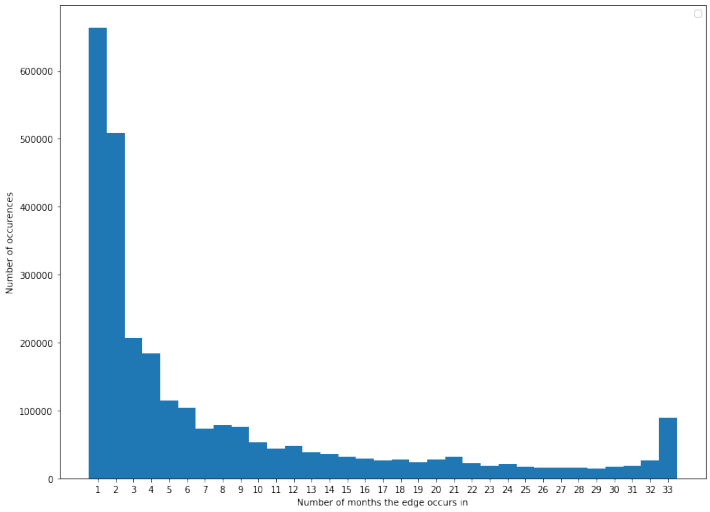
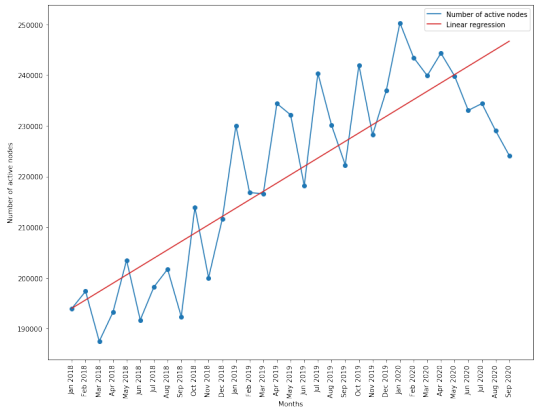


Figure 3.6: Histogram depicting how many months an edge typically occurs in.

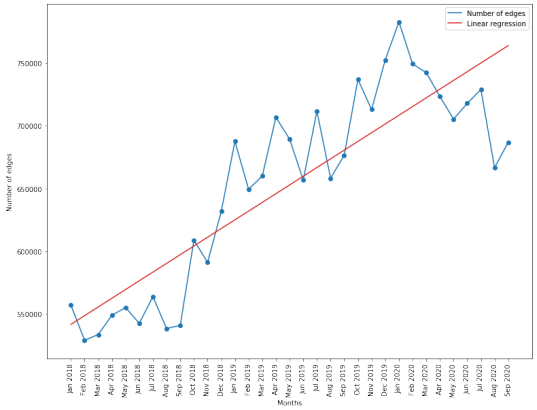
3.4.1. Is the size of the network increasing over time?

Intuitively, the size of the network should be increasing over time, since Exact is a growing company that is attracting more and more users every month which should result in a growth in the number of active nodes in the network. Additionally, the economy is usually growing which should result in an increasing number of edges. To answer the first question of how the size of the network is behaving over time, we can plot the number of active nodes and the number of edges per month. These plots can be found in figure 3.7.

In figures 3.7a and 3.7b, we can see the number of active nodes and number of edges in the dynamic network for each month between January 2018 and September 2020. The red lines are least-squares linear regression lines that minimize the residual sum of squares to fit the data points. In figure 3.7a we can see that the number of active nodes, being nodes with a degree > 0, is generally increasing over time. This is confirmed by the linear regression line that has a positive slope, indicating that there generally is an increase in the number of active nodes over time. The same holds for the number of edges in the network as depicted in figure 3.7b, where we can see that generally, the number of edges in the network is increasing, which is confirmed by the positive slope of the linear regression line.



(a) Number of active nodes per month between January 2018 and September 2020.



(b) Number of edges per month between January 2018 and September 2020.

Figure 3.7: Figures depicting the number of active nodes and edges per month between January 2018 and September 2020.

3.4.2. Are closer months in the network more similar to each other than months that are far apart?

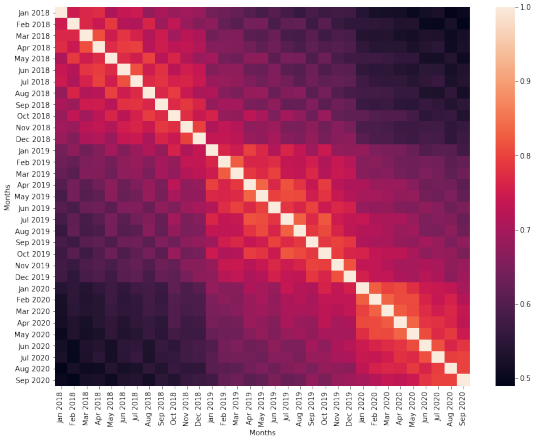
When reasoning about the dynamic network and how it is continuously evolving, it is intuitive that somehow closer months in the network are should be generally similar to each other than months that are further apart since the odds that the same companies are active in consecutive months are higher than the odds that the same companies are active in months that are far apart. To answer this question, we have to compare all months in the network to each other and verify whether closer months are more similar. Since we have 33 months worth of data, we will have to make $\sum_{n=1}^{33} n = 561$ comparisons, making it important that these comparisons can be performed rather quickly. Because various definitions and measures of network similarity exist, we will apply several graph similarity metrics that allow us to verify what types of similarity closer months are more similar than months that are further apart.

The first network similarity metric that we will be discussing is the Jaccard similarity as discussed in section 2.2 between the sets of active nodes. These results can be found in figure 3.8. In figure 3.8a, we can see a heatmap containing the Jaccard similarities between the sets of active nodes for all months. On the diagonal, the similarities are 1 since we are comparing the set of nodes against itself. Intuitively, we want squares close to the diagonal to have a higher value than squares that are further away since months that are close to each other will be close to the diagonal. In figure 3.8a, this seems to be the case. However, to get a more conclusive answer, we plot the average Jaccard similarity between the sets of active nodes per number of months between compared graphs in figure 3.8b. For example, if the distance between months is 1, we compare January 2018 to February 2018, February 2018 to March 2018, and so forth. When the distance between months is 2, we compare January 2018 to March 2018, February 2018 to April 2018, and so forth. In this figure, we see that in general, the larger the number of months between months we are comparing, the lower the average Jaccard similarity between the set of active nodes is, meaning that in terms of Jaccard similarity between sets of active nodes, closer months tend to be more similar than months that are further away.

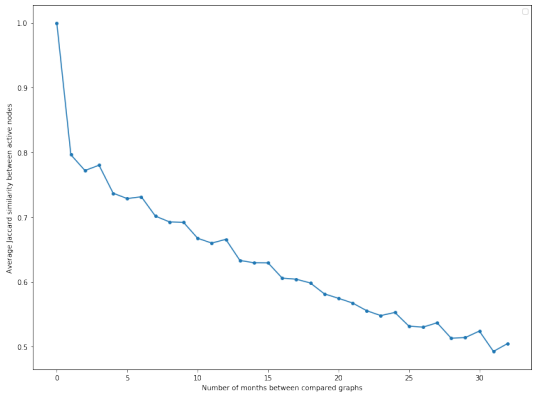
Similarly, we can plot Jaccard similarity between the sets of edges for all months in figure 3.9b. Similar to the Jaccard similarity for the sets of active nodes, the heatmap of the Jaccard similarity of edges as found in figure 3.9a seems to indicate that closer months are indeed more similar in terms of Jaccard similarity of sets of edges. This is confirmed by figure 3.9b, where we can see the Jaccard similarity between sets of edges decrease as the number of months between compared graphs becomes larger.

Alternatively, we can also use the Jensen-Shannon divergence of degree distributions as described in [8] as a measure of distance between graphs based on the dissimilarity of their degree distributions. We can then measure the similarity between two graphs as $1-j$, where j is the Jensen-Shannon divergence of degree distributions of two graphs. In short, we use the degree occurrences of the two networks to derive two probability distributions. We then calculate the Jensen-Shannon divergence, which is a smoothed and symmetric version of the Kullback–Leibler divergence, as such:

$$JSD(P,Q) = \frac{1}{2}KL(P,\frac{1}{2}(P+Q)) + \frac{1}{2}KL(Q,\frac{1}{2}(P+Q))$$

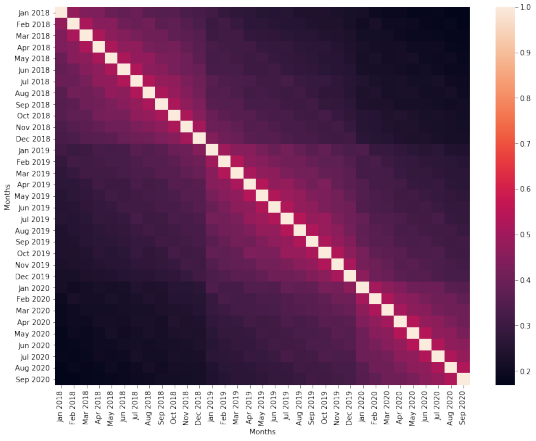


(a) Jaccard similarity between sets of active nodes.

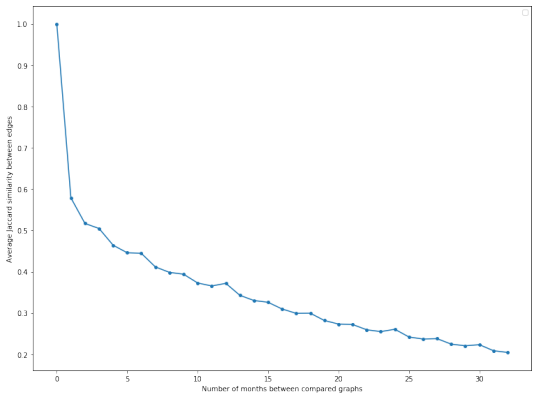


(b) Average Jaccard similarity between sets of active nodes per number of months between compared graphs.

Figure 3.8: Figures depicting Jaccard similarity between sets of active nodes.



(a) Jaccard similarity between sets of edges.



(b) Average Jaccard similarity between sets of edges per number of months between compared graphs.

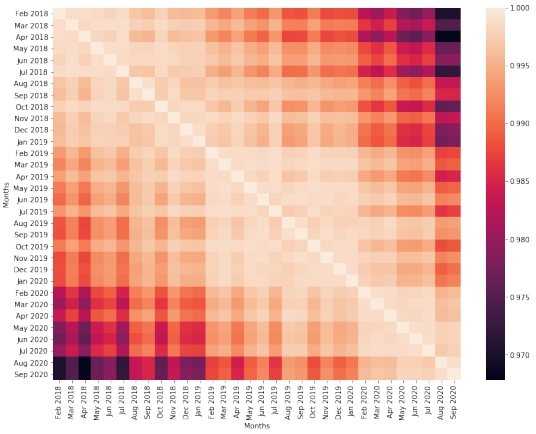
Figure 3.9: Figures depicting Jaccard similarity between sets of edges.

, where $KL(P,Q)$ is the Kullback-Leibler divergence of two distributions and is calculated as:

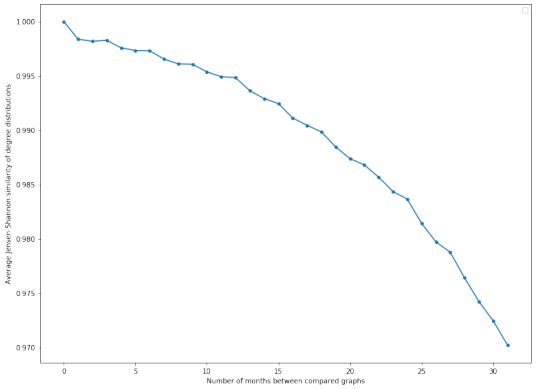
$$KL(P,Q) = \sum_{x \in X} P(x) log(\frac{P(x)}{Q(x)})$$

To be able to properly capture the topological changes in the network isolated from the growth of the set of nodes, we only consider the set of starting nodes, being the set of nodes that had a degree > 0 in January 2018, for each network. Since nodes in this set of starting nodes are heavily overrepresented in January 2018, we remove this month from the comparison. Plots containing this similarity measurement can be found in figure 3.10. In the heatmap depicted in figure 3.10a, we can see that squares closer to the diagonal tend to have a higher similarity than squares that are further away, meaning that when we compare months that are closer together, the distribution of their degrees tends to be more similar. This is confirmed by figure 3.10b, where we can see that when the number of months between compared graphs increases, the Jensen-Shannon similarity of degree distributions between them decreases.

Thirdly, to try to assess the similarity between months in terms of connectivity, we will take a look at the DeltaCon distance [16] as discussed in section 2.2 between all months. Similar to the Jensen-Shannon similarity of degree distributions as described above, we only use the set of starting nodes to be able to properly capture the connectivity changes in the networks. Due to memory and runtime constraints, graph sampling had to be performed to be able to calculate the DeltaCon distances for the Cartesian product of the set of all months in a feasible time. The authors of [20] discuss several approaches to graph sampling, and conclude that "simple uniform random node selection performs surprisingly well". Therefore, we decided to sample a



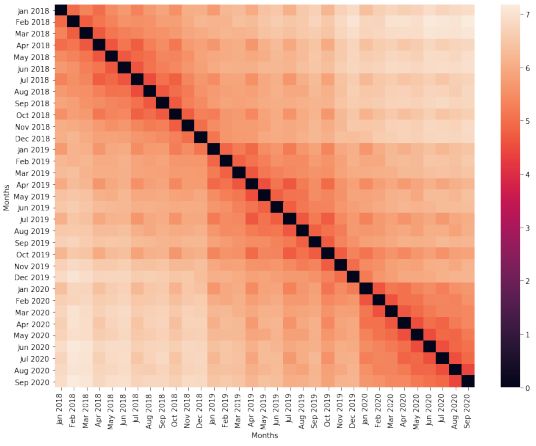
(a) Jensen-Shannon similarity of degree distributions between starting nodes of months.



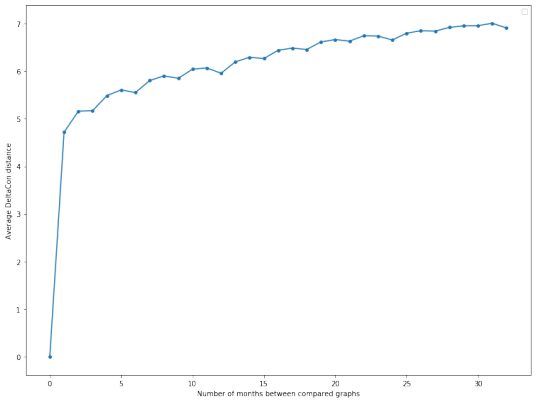
(b) Average Jensen-Shannon similarity of degree distributions per number of months between compared graphs.

Figure 3.10: Figures depicting the Jensen-Shannon similarity of degree distributions.

uniform random set of 10 000 nodes for all graphs 5 times and use the average DeltaCon distances of these 5 samplings to decrease the variance. The results of this experiment are depicted in figure 3.11. As we can see in the heatmap in figure 3.11a, we can see that months that are closer together tend to have a lower DeltaCon distance between them, meaning that months that are closer together tend to have more similar node influences than those of months that are further apart. This is confirmed by the line graph in figure 3.11b, where we can see that the DeltaCon distance steadily increases when the number of months between compared graphs increases.



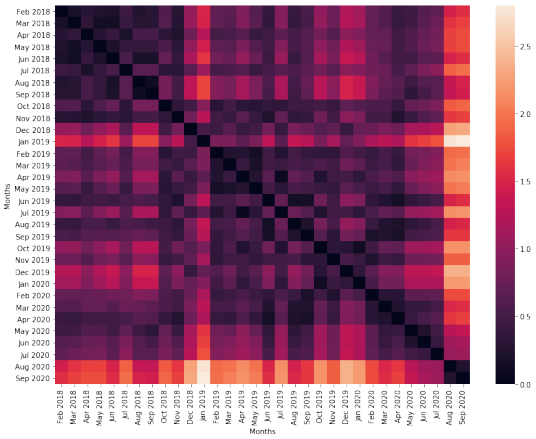
(a) DeltaCon distance between months.



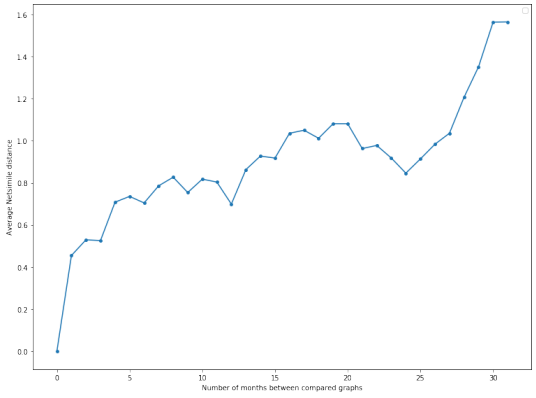
(b) Average DeltaCon distance per number of months between compared graphs.

Figure 3.11: Figures depicting the DeltaCon distance.

Lastly, we will take a look at the NetSimile distance [5] as discussed in 2.2 between graphs to assess the topological similarity of these networks. Similar to the Jensen-Shannon similarity of degree distributions and the DeltaCon distance calculations as described above, we only use the set of starting nodes to be able to properly capture the topological changes in the networks. The results of this experiment can be found in figure 3.12. In the heatmap depicted in figure 3.12a, it is difficult to see whether squares close to the diagonal have a significantly lower distance than squares that are further away. Therefore, we take a look at the line graph in figure 3.12b. Here, we can see that although the average NetSimile distance seems to increase when the number of months between compared graphs increases, the relation does seem to be noisier with regard to the distance metrics as discussed above.



(a) NetSimile distance between months.

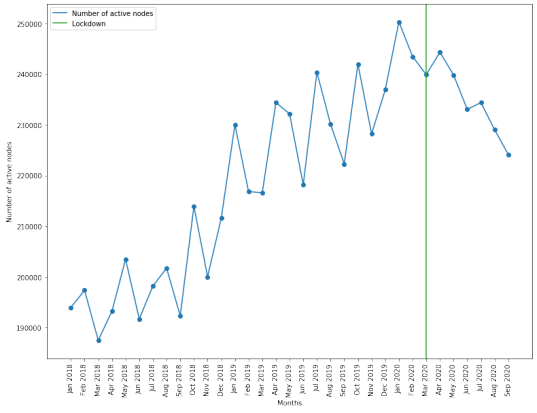


(b) Average NetSimile distance per number of months between compared graphs.

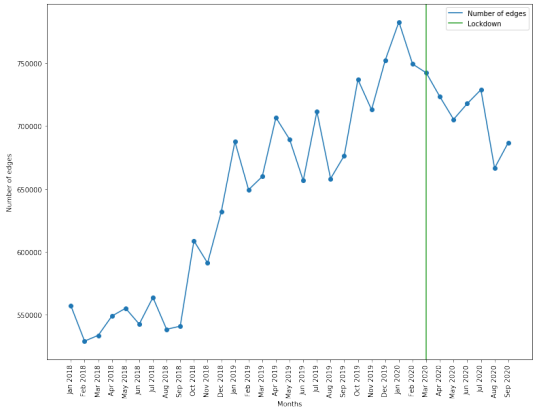
Figure 3.12: Figures depicting the NetSimile distance.

3.4.3. Can we see the effect of COVID-19 in some metrics in the dynamic network?

For our dynamic network to be a useful way of preparing the data, the effects of the COVID-19 lockdown have to be visible in some way in the dynamic network. To assess whether this is the case, we will first plot the number of active nodes and the number of edges in the network over time, together with an indicator of when the first COVID-19 lockdown in The Netherlands happened. These plots can be found in figure 3.13.



(a) Number of active nodes per month, with a lockdown indicator on March 2020.



(b) Number of edges per month, with a lockdown indicator on March 2020.

Figure 3.13: Figures depicting the number of active nodes and edges per month, with a lockdown indicator on March 2020.

In figures 3.13a and 3.13b, we can see that the number of active nodes and edges both decline after the COVID-19 lockdown, whereas they were both increasing before the lockdown as discussed in section 3.4.1. However, the question remains whether this decrease in active nodes and edges is due to either a decrease in the number of Exact users or whether the nodes in the network are becoming less active. To gain some more insight, the number of type D accounts, i.e. the number of Exact users, over time has been added to the plot containing the number of active nodes in figure 3.14. In this figure, we can see that the number of type D accounts is increasing over time, similarly to the number of active nodes in the network. However, we can also see that the decrease in the number of active nodes after the COVID-19 lockdown is not reflected in the number of type D accounts, leading us to believe that the decrease in the number of active nodes is mostly happening because the nodes in the network are generally less active.

To further assess the assumption that the number of active nodes is decreasing because nodes are generally less active, we took the set of starting nodes, being the set of nodes that had a degree > 0 in January 2018, and used this set to look at this subgraph over time. In figure 3.15, the average degree and its standard deviation of this subgraph are shown. In this figure, we see that before the lockdown, the average degree tends to be increasing apart from the large drop from January 2018 to February 2018, which we should not

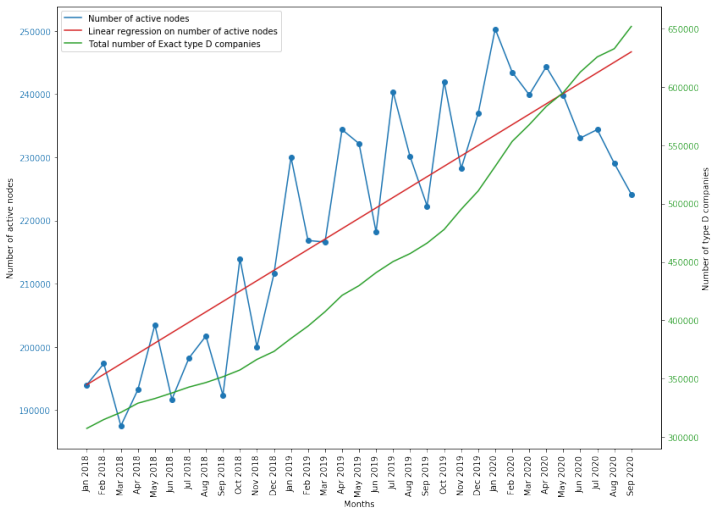


Figure 3.14: Number of active nodes per month with the number of type D accounts per month.

reach too much into since we picked all nodes that have a degree > 0 in January 2018, making the number of edges overrepresented in January 2018. We can also see that the average degree of the set of starting nodes is decreasing after the COVID-19 lockdown in March 2020, further leading us to believe that the decrease in the number of active nodes and edges to be a result of the nodes in the network being less active as a result of the lockdown. The standard deviation of the degrees tends to follow a very similar trend as the average degree. Because the degree follows a power-law distribution as depicted in figure 3.4, we believe that the standard deviation follows a similar trend to the average because that when the average degree decreases, the standard deviation also decreases since a large number of samples in our dataset are getting closer to the average. The same goes for when the average degree increases; a large number of samples in our dataset are getting further away from the average.

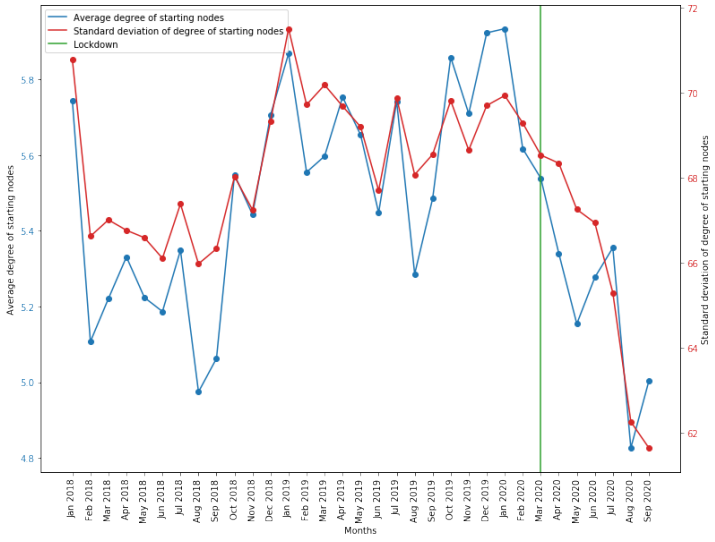


Figure 3.15: Average degree and standard deviation of degree of the subgraph of starting nodes per month.

Similarly, we can plot the average clustering coefficient for the subgraph of starting nodes over time. The results can be found in figure 3.16. The average clustering coefficient C is calculated as:

$$C = \frac{1}{n} \sum_{v \in G} c_v$$

, where n is the number of nodes, G is the graph, and c_v is calculated by taking the direct neighbors of v

and taking the number of edges between them divided by the total number of possible edges between these neighbors. In figure 3.16, we can see that the average clustering coefficient fluctuates between approximately 0.050 and 0.060 before the lockdown. However, after the lockdown, the average clustering coefficient slightly drops to 0.044, further leaving us to believe that the effects from the lockdown are visible in the behavior of the nodes in the network.

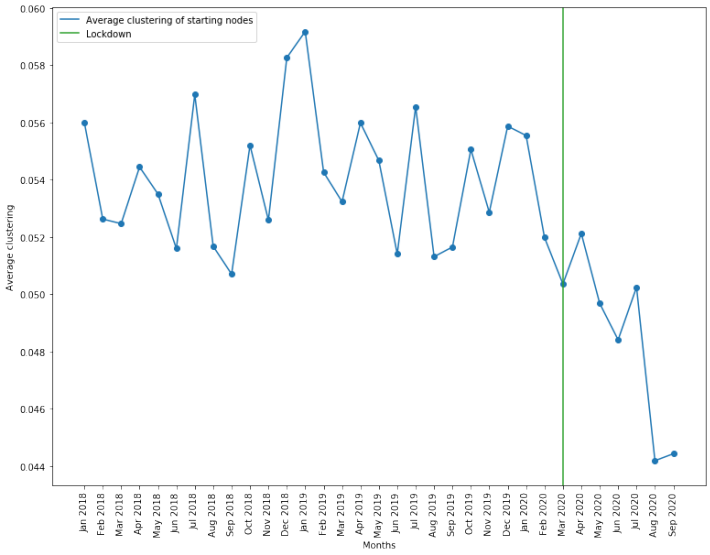


Figure 3.16: Clustering coefficient of the subgraph of starting nodes per month.

However, there are multiple reasons why this drop in the clustering coefficient could be happening in the network. On one hand, a decrease in the number of links will naturally lead to a decrease in the clustering coefficient, but on the other hand, the clustering coefficient could be dropping because important links are disappearing. To verify whether links are disappearing homogeneously, we plot the mean clustering coefficient of the set of starting nodes similar to the plot in figure 3.16 together with the link density in figure 3.17a. The link density is defined as the fraction of edges with regard to the total number of possible edges and is calculated as:

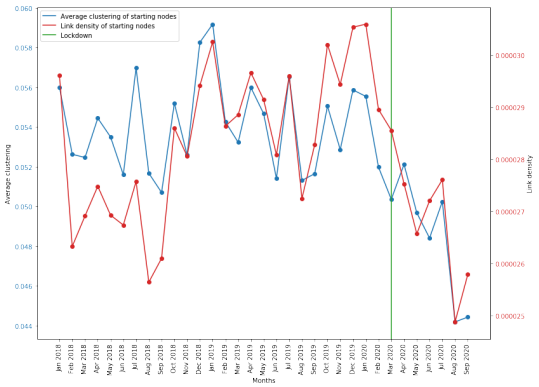
$$d = \frac{e}{\frac{n(n-1)}{2}}$$

, where d is the link density, e is the number of edges in the network and n is the number of nodes in the network. In figure 3.17a, we can see that the clustering coefficient and link density tend to follow a very similar trend, leading us to believe that after the coronavirus lockdown, edges tend to disappear homogeneously. This claim is further backed up by figure 3.17b, where the degree assortativity of the subgraph of starting nodes per month is plotted. This degree assortativity measures whether nodes with a similar degree tend to be connected. The assortativity ranges between -1 and 1, where -1 indicates that nodes with a similar degree tend to not be connected, whereas a value of 1 indicates that nodes with a similar degree tend to be connected. As we can see in figure 3.17b, the degree assortativity does not necessarily change significantly after the lockdown, further leading us to believe that edges are disappearing homogeneously.

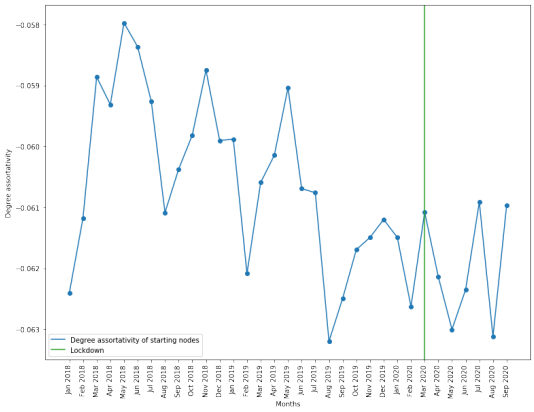
3.4.4. Are there significant differences in network features between sectors?

Lastly, we are also curious as to whether significant differences exist between various sectors, and whether they are impacted differently by the COVID-19 lockdown in March 2020. To start, we will take a look at the distribution of sector occurrences in our dynamic network as depicted in figure 3.18. In this figure, we can see that over 140 000 companies do not have a sector code. The meanings of these sector codes can be found in appendix A. Additionally, we also see that a large number of sectors contain a small number of nodes.

Next, we will take a look at the number of edges between nodes of the same sector versus the number of edges versus nodes from different sectors as depicted in figure 3.19 to see whether there is a difference in trend between these two distinct sets of edges. In figure 3.19a, all nodes are considered, whereas in figure 3.19b, only starting nodes are considered. In both of these figures, we can see that there does not seem to



(a) Average clustering coefficient and average link density of the subgraph of starting nodes per month.



(b) Degree assortativity of the subgraph of starting nodes per month.

Figure 3.17: Figures verifying whether links are disappearing homogeneously.

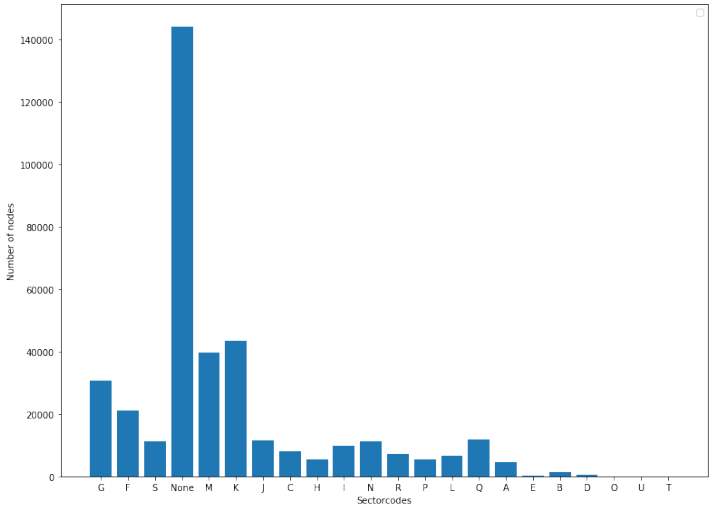
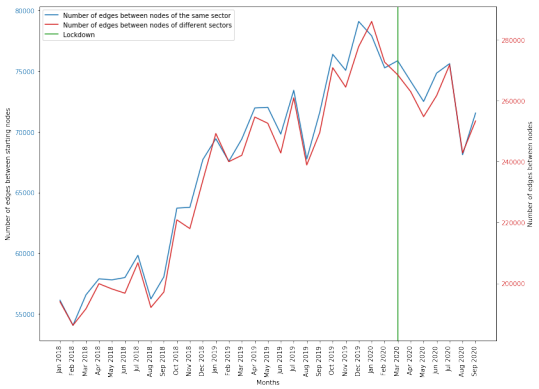
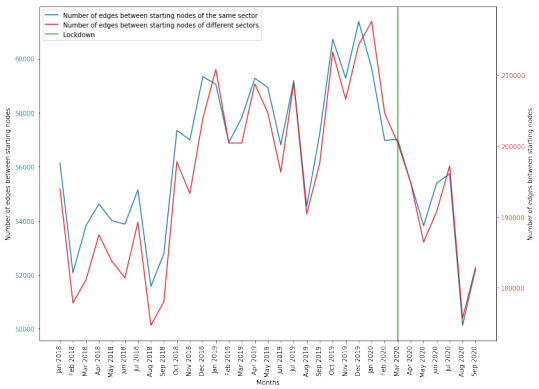


Figure 3.18: Sector occurrences in the dynamic network.

be a significant difference between the trend in the number of edges of nodes of the same sector or nodes of different sectors.



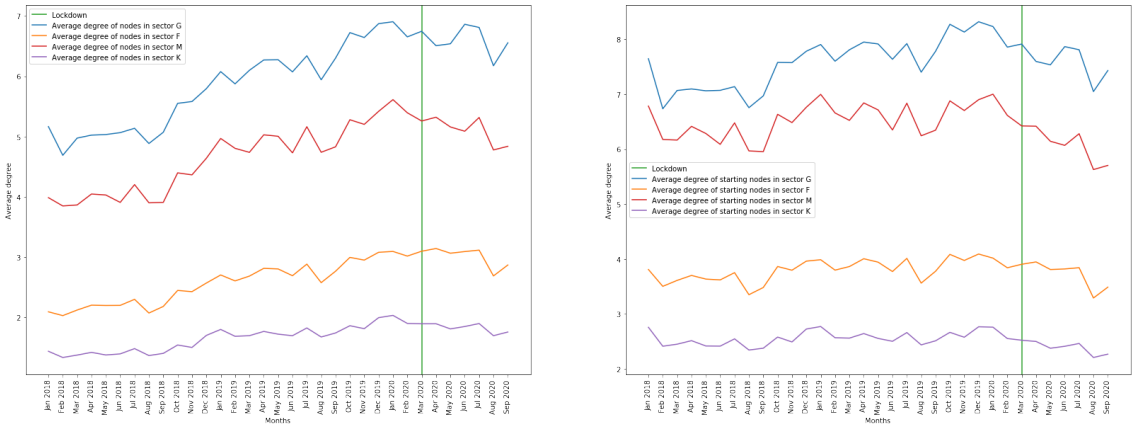
(a) Number of edges between nodes of the same sector and between nodes of different sectors.



(b) Number of edges between starting nodes of the same sector and between nodes of different sectors.

Figure 3.19: Figures depicting the number of edges between nodes of the same sector and between nodes of different sectors.

To further assess whether there exist significant differences between nodes in different sectors, we plot the average degree of all nodes per sector in figure 3.20. To ensure that the results are sufficiently reliable we only consider sectors with at least 20 000 nodes, being sectors G, F, M, and K. In figure 3.20a, all nodes are considered, whereas in figure 3.20b, only starting nodes are considered. In both these plots, we see that the average degree per sector varies significantly, leading us to believe that there are significant differences in network features between sectors. We also see that the average degree of all sectors seems to be decreasing as a result of the COVID-19 lockdown in The Netherlands, but there seem to be differences in how heavily this decrease in average degree is per sector.



(a) Average degree of nodes per sector.

(b) Average degree of starting nodes per sector.

Figure 3.20: Figures depicting the average degree of nodes in different sectors.

4

SQ2: Improving KPI predictions using node features

In this chapter, we will be looking at the second subquestion of how to utilize our dynamic network to improve existing KPI forecasting techniques. First, we discuss the influence of the COVID-19 on KPIs in section 4.1. Then, we will discuss the experimental setup for KPI forecasting using node features in section 4.2. Afterward, we will execute the experiments as described in section 4.2 for various KPIs, namely Revenue in section 4.3, CashFlowMonthly in section 4.4, and D2C_14d in section 4.5.

4.1. COVID-19 lockdown influence on KPIs

In section 3.4.3, we have verified that the effects of the COVID-19 lockdown in The Netherlands are visible in some features of our dynamic network. Additionally, we still need to verify whether the effect of the coronavirus lockdown is also visible in certain characteristics of the companies that the dynamic network captures. To assess the influence of the COVID-19 lockdown on Dutch SMEs, we will take a look at several Key Performance Indicators, or KPIs for short. These KPIs are measurements that indicate whether a company is doing well at a certain moment in time. At Exact, these KPIs are calculated on a monthly basis. Not all of these KPIs are computed for every company. The possible KPIs that a company in the Exact dataset can have are:

- **Cost**. Calculated as the sum of all costs made by the company over this month.
- **Revenue**. Calculated as the sum of all incomes received by the company over this month.
- **CashFlowMonthly**. Calculated as $revenue - cost$ over this month.
- **CashFlowPosition**. Calculated as $revenue - cost$ cumulative from the beginning up until this month.
- **D2C_14d**. Calculated as the average payment time for paid sales invoices with a due date of 14 days calculated over this month.
- **D2C_30d**. Calculated as the average payment time for paid sales invoices with a due date of 30 days calculated over this month.
- **D2C_all**. Calculated as the average payment time for all paid sales invoices calculated over this month.

An overview of statistics on how many companies have a certain KPI computed for them can be found in figure 4.1. In this figure, we can see that there are over 350 000 nodes in the network. Of these nodes, about 250 000 nodes have KPIs available for Cost, Revenue, CashFlowMonthly, and CashFlowPosition. A smaller amount of nodes have computed values for D2C_all, D2C_14d, and D2C_30d.

To assess whether the effect of the COVID-19 lockdown is visible in the companies that the network captures, we take a look at the changes in these KPIs over time. However, there are several ways to approach this. Firstly, we could look at the average value of the KPIs per month. However, the problem with this approach is that the variance in the scale of these KPIs is often large. Since SMEs can vary greatly in size, relatively large

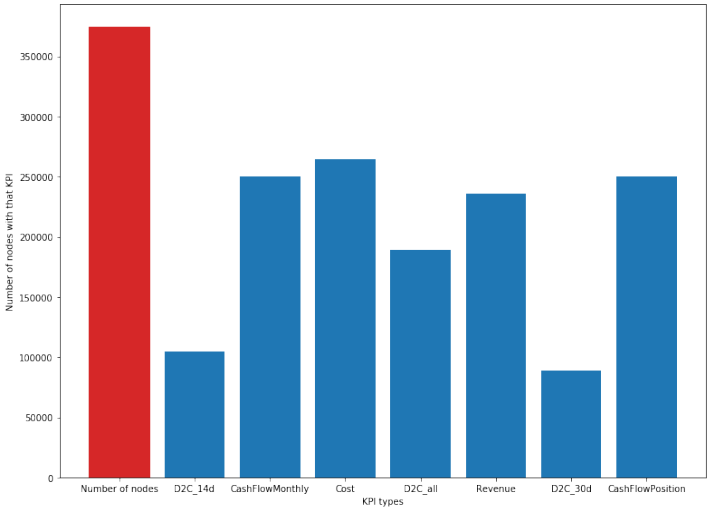
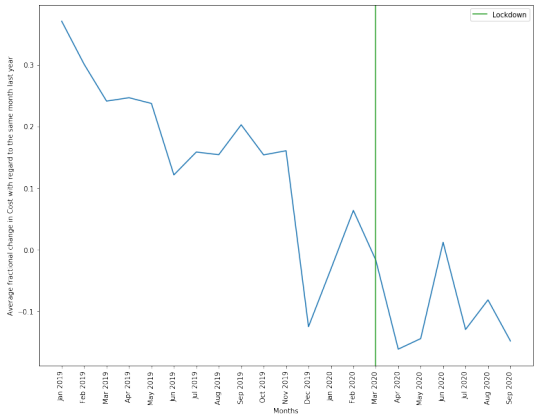
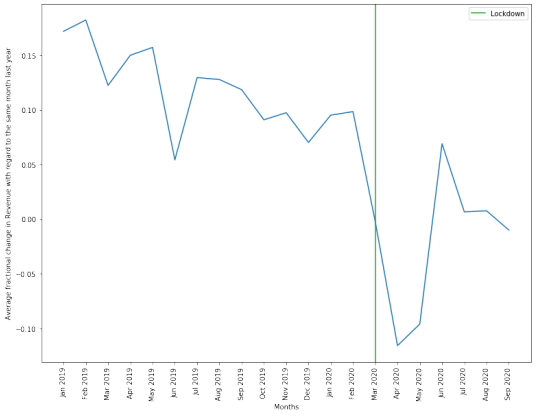


Figure 4.1: Number of nodes for which certain KPIs are calculated.

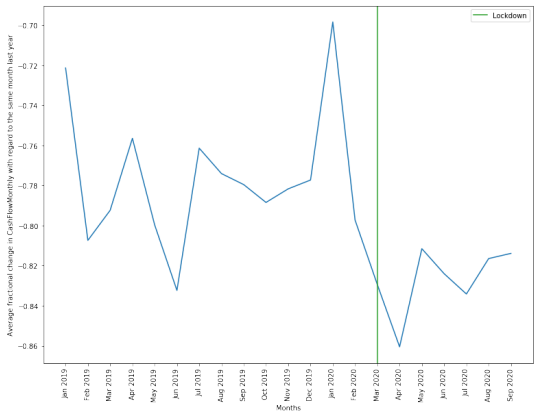
changes for smaller companies will have a very small effect on the average value of certain KPIs per month. This is not a desired property since we are interested in the effect of the COVID-19 lockdown on all SMEs, regardless of the size of those SMEs. Therefore, to solve this issue, we could look at the average relative change for each company with regard to last month. However. the problem with this approach is that there is a lot of seasonality in the data which makes it unreliable to draw sensible conclusions from comparing consecutive months. To solve this problem, we will consider the average relative change in KPIs for each company with regard to the same month in the previous year. These changes for all KPIs over time can be found in figure 4.2. In figure 4.2a, we can see the average relative change in Cost for all companies with regard to the same month previous year. In this figure, we see that there does seem to be a decrease around the coronavirus lockdown, but it is difficult to attribute this decrease to the lockdown since we see multiple similar decreases outside of this lockdown period. This is different for the Revenue as depicted in figure 4.2b, where we see a very large unprecedented decrease in Revenue around the COVID-19 lockdown. In figure 4.2c we can see the average relative change for CashFlowMonthly with regard to the same month previous year. In this figure, we can see that there seems to be a significant drop in CashFlowMonthly around the coronavirus lockdown. This cannot be said for the CashFlowPosition in figure 4.2d, where we can see that the CashFlowPosition is trending upwards very quickly after the lockdown. D2C_all, D2C_30d and D2C_14d as depicted in figures 4.2e, 4.2f and 4.2g respectively all seem to follow similar trends where after the coronavirus lockdown, there seems to be a slight increase in average payment time before decreasing again. In terms of Revenue, CashFlowMonthly, and the various D2C KPIs, we believe that the result of the coronavirus lockdown is clearly visible for the companies of the Exact dataset.



(a) Average relative change in Cost for all companies with regard to the same month previous year.

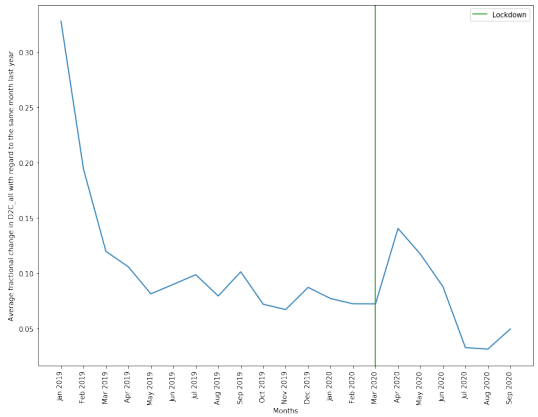


(b) Average relative change in Revenue for all companies with regard to the same month previous year.

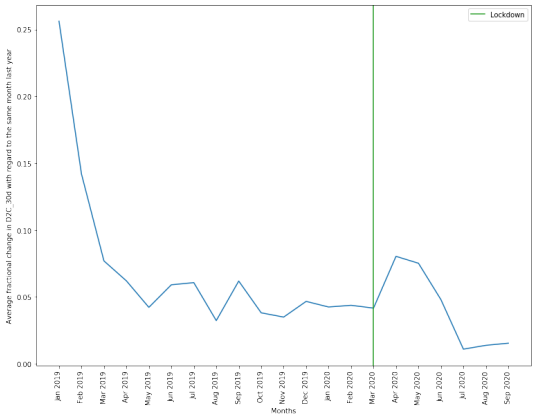


(c) Average relative change in CashFlowMonthly for all companies with regard to the same month previous year.

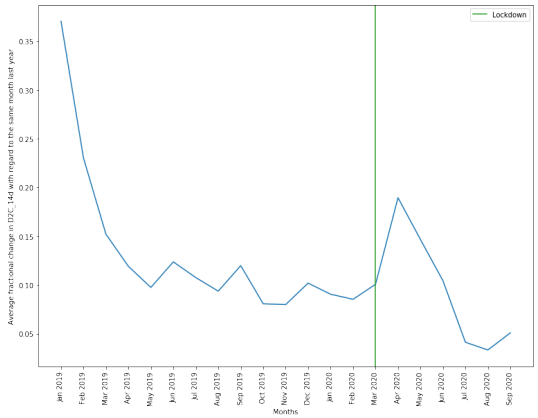
(d) Average relative change in CashFlowPosition for all companies with regard to the same month previous year.



(e) Average relative change in D2C_all for all companies with regard to the same month previous year.



(f) Average relative change in D2C_30d for all companies with regard to the same month previous year.



(g) Average relative change in D2C_14d for all companies with regard to the same month previous year.

Figure 4.2: Figures depicting the average relative change in KPIs for all companies with regard to the same month previous year.

Since we can see the effect of the coronavirus lockdown in the network as discussed in section 3.4.3, and we can also see the effect of the coronavirus lockdown in The Netherlands for Revenue, CashFlowMonthly, and the various D2C KPIs, we will try to improve KPI forecasts for Revenue, CashFlowMonthly, and one of the D2C KPIs, namely D2C_14d, using node features. These KPIs are very important for SMEs; generating revenue is essential for SMEs for obvious reasons, just like having a steady cash flow. Also, getting your invoices paid on time is another very important aspect for SMEs since having money to invest is crucial. Improving these KPI predictions is valuable for Exact since it employs KPI forecasting for its customers. Improving the quality of these forecasts delivers more value to the customers of Exact, and is therefore valuable for Exact itself. For the scientific community, this work provides an analysis of the usefulness of node features for KPI forecasting.

4.2. Experimental setup for KPI forecasting

In this section, we will discuss the setup for the experiments that we are going to use for KPI forecasting. These experiments will be executed for Revenue in section 4.3, CashFlowMonthly in section 4.4, and D2C_14d in section 4.5.

For our experiments, we will try to predict the KPI of companies for April 2020 using a linear regression model. As we can see in the figures in figure 4.2, we can see that in April 2020, for some KPIs, some unexpected changes occurred, hence the choice to use this month as our prediction target. We hypothesize that because of the sudden change in the values of these KPIs that is not reflected in the KPI values before as a result of the coronavirus lockdown, enriching the predictions using node features could increase the accuracy of KPI predictions. Linear regression models have often been used for KPI prediction. The authors of [3] found that, in the domain of movie revenue prediction, linear regressions are the most widely used machine learning model for revenue prediction. This, in combination with the fact that the linear regression model has the advantage that it is very explainable, is the reason why we will use a linear regression model for KPI prediction in this thesis. To check whether node features can be used to improve KPI forecasting, we train various KPI forecasting models and vary the data the linear regression models have access to. We will experiment using the following KPI forecasting models:

- **Baseline**. As the baseline model, we will use a linear regression model that minimizes the residual sum of squares. This model has access to the 12 previous monthly KPI values of the company and does not use any node features.

- **Degree**. For this approach, we will use a similar linear regression model. However, this model not only has access to the 12 previous monthly KPI values, but we will also provide the model with the 12 corresponding previous degrees of the node of that company.

- **Clustering coefficient**. For this approach, we will again use a linear regression model. However, this model not only has access to the 12 previous monthly KPI values, but we will also provide the model with the 12 corresponding previous clustering coefficients of the node of that company.

- **node2vec**. For this approach, we will again use a linear regression model. However, this model not only has access to the 12 previous monthly KPI values, but we will also provide the model with the 12 corresponding previous node2vec embeddings of the node of that company. This node2vec embedding is generated by an implementation as described in [1], where the author uses rejection sampling to reduce the computational complexity from linear time to logarithmic time. Since the author of [14] found that the parameters do not make a large difference on a similar dataset by Exact, we will use the parameters as mentioned in [14], being $num\_walks = 20$, $embedding\_size = 20$, $walk\_length = 40$, $p = 2$, and $q = 0.25$.

- **Degree + clustering**. For this approach, we will again use a linear regression model. However, this model not only has access to the 12 previous monthly KPI values, but we will also provide the model with the 12 corresponding previous degrees and clustering coefficients of the node of that company.

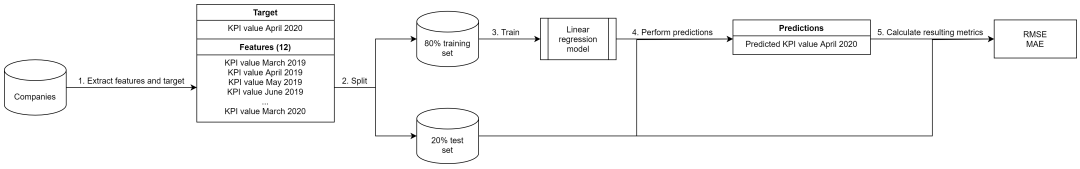
- **Degree + node2vec**. For this approach, we will again use a linear regression model. However, this model not only has access to the 12 previous monthly KPI values, but we will also provide the model with the 12 corresponding previous degrees and the 12 corresponding previous node2vec embeddings of the node of that company.

- **Clustering + node2vec**. For this approach, we will again use a linear regression model. However, this model not only has access to the 12 previous monthly KPI values, but we will also provide the model with the 12 corresponding previous clustering coefficients and the 12 corresponding previous node2vec embeddings of the node of that company.
- **Degree + clustering + node2vec**. For this approach, we will again use a linear regression model. However, this model not only has access to the 12 previous monthly KPI values, but we will also provide the model with the 12 corresponding previous degrees, the 12 corresponding previous clustering coefficients, and the 12 corresponding previous node2vec embeddings of the node of that company.

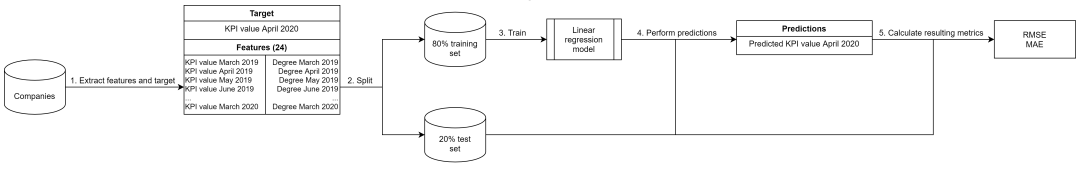
We will score the models on the following metrics:

- **Mean absolute error**, or MAE for short. It is calculated by: $MAE = \frac{\sum_{i=1}^{n} |Y_i - \hat{Y}_i|}{n}$, where n is the amount of predictions, Y_i is the actual value being predicted and $\hat{Y}_i$ is the value of the prediction. Simply put, it is the average difference between the predicted value and the actual value.
- **Root mean squared error**, or RMSE for short. It is calculated by: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$, where n is the amount of predictions, Y_i is the actual value being predicted and $\hat{Y}_i$ is the value of the prediction. Since RMSE squares the error, it assigns more weight to larger errors compared to MAE.

An overview of the design of the baseline experiment and the degree experiment, as an example for all the node feature experiments, is shown in figure 4.3. In figure 4.3a, the experimental design of the baseline model is shown. As we can see, we start with a dataset of companies. For our first step, we extract the 12 features, being the KPI values between March 2019 and March 2020, and the target, being the KPI value for April 2020, for all companies. Secondly, we randomly split up our dataset into a training set containing 80% and a test set containing 20% of the companies. The third step consists of training the linear regression model using the features and the target from the training set. Fourthly, we perform our predictions on the test set using the test set features, resulting in a set of predictions that we will utilize in our fifth and final step, where we compare our predictions against the target KPI values from the test set, ultimately calculating the RMSE and MAE of our predictions. When we compare the baseline experiment design in figure 4.3a against the degree experiment design in figure 4.3b, we can see that the only difference occurs in step 1. For the degree experiment, we extract 24 features in comparison to the 12 that are extracted for the baseline model. Not only do we extract the 12 KPI values between March 2019 and March 2020, but we also extract the node degree for the company between March 2019 and March 2020, resulting in 24 features.



(a) Experimental design of the baseline model.



(b) Experimental design of the degree model.

Figure 4.3: Experimental design of the baseline model and the degree model.

However, before we perform the predictions, we will start by performing a pre-analysis by taking a look at the distribution of the values of the KPI. Next, we will check whether there exists a correlation between the node features and the KPI that we are predicting using a plot containing a kernel density estimate together with a linear regression line. To assess the potential usability of the node2vec embeddings, we use t-SNE dimensionality reduction as described in [36] to verify a possible correlation between the KPI and the node2vec embeddings. After the pre-analysis, we will present the results of the various KPI forecasting models. After examining the results, we will perform a statistical analysis on the baseline model and the best enriched

model. Then, we will perform a post-analysis by diving deeper into the differences between the best model that has been enriched with node features and the baseline model by splitting up the dataset with regard to sector, mean KPI value, and mean degree, and then train and predict using these split datasets as input to the experiments as described in figure 4.3 to examine if there is a certain subset of data where the enriched KPI forecasting model outperforms the baseline further. Lastly, we will try to verify the hypothesis where we think that for April 2020, historical data is less useful, and thus enriching our predictions with node features is more useful by performing predictions for a month before the coronavirus lockdown, the first month where the coronavirus lockdown is visible, and a month deeper in the COVID-19 pandemic. We chose the month before the coronavirus lockdown to be September 2019. The first month where the coronavirus lockdown is visible is April 2020 as discussed above, and as a month deeper in the COVID-19 pandemic, we chose September 2020 as it is the last available month in our dataset. In these post-analysis experiments, we chose to do the training of the model on the corresponding subset of data only instead of using the trained model on all data before doing predictions for the corresponding subset of data because often the variance in the KPIs to predict of all companies is large, so splitting up the dataset could potentially increase performance significantly. This is also done for existing KPI predictions techniques at Exact.

4.3. Revenue forecasting

In this section, we will take a look at Revenue forecasting and how to improve these forecasts using node features using the experiments as described in section 4.2.

4.3.1. Pre-analysis

To see whether we can use node features to improve Revenue predictions, we will first take a look at the Revenue distribution in figure 4.4. In this histogram, we can see that the most occurring Revenue is between 0 and 1000, and the amount of occurrences is generally decreasing as the Revenue gets higher.

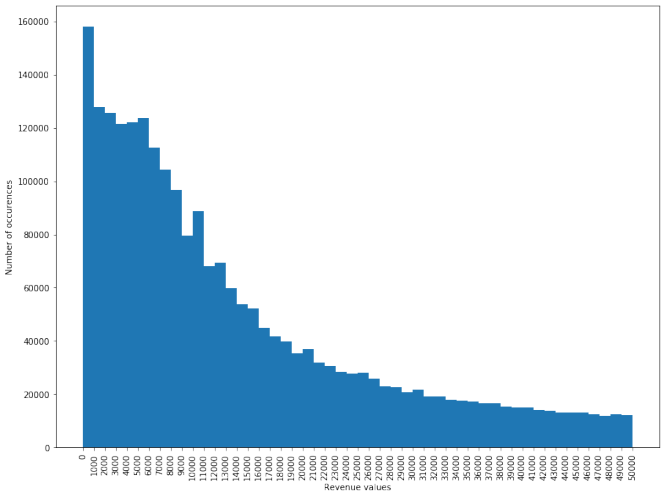
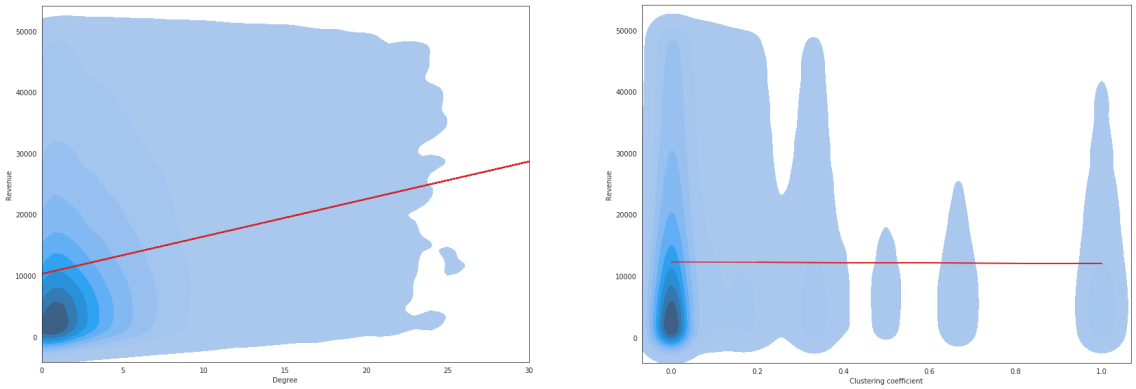


Figure 4.4: Histogram depicting the Revenue distribution.

Additionally, we are also interested in whether there is a correlation between the node features and Revenue. Kernel density estimations of these node features and Revenue are depicted in figure 4.5, together with a linear regression line that minimizes the residual sum of squares indicating the correlation between the node features and Revenue. In figure 4.5a, we see that the degree is often close to zero, which is in line with earlier findings as reported in figure 3.4. In figure 4.5a, we also see that there seems to be a positive correlation between degree and Revenue. The Pearson correlation coefficient between Revenue and degree, calculated as $\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2\sum(y_i-\bar{y})^2}}$, is 0.27, leading us to believe that degree could potentially be a suitable node feature to improve Revenue predictions with. In figure 4.5b, we see that a lot of the clustering coefficients of the nodes also tend to be close to zero. The linear regression line does not indicate a clear correlation between clustering coefficient and Revenue, leading us to believe that the clustering coefficient of a node may be a less suitable node feature to improve Revenue predictions with.



(a) Density plot and linear regression line indicating the correlation between degree and Revenue.

(b) Density plot and linear regression line indicating the correlation between clustering coefficient and Revenue.

Figure 4.5: Density plots and linear regression lines indicating the correlation between various node features and Revenue.

To assess the potential usability of the node2vec embeddings, we use t-SNE dimensionality reduction as described in [36] to further verify a possible correlation between Revenue and the node2vec embeddings. These results can be found in figure 4.6. In this figure, we see that we used t-SNE to reduce the node2vec embeddings to two dimensions. All data points also have a color indicating the Revenue of that node. In this figure, we cannot clearly see clusters of node2vec embeddings with similar Revenue. This could mean that the node2vec embeddings might not be suitable for Revenue prediction. However, since we were required to perform dimensionality reduction on the node2vec embeddings for visualization purposes, it could be the case that providing the actual full node2vec embeddings could either improve or decrease the performance.

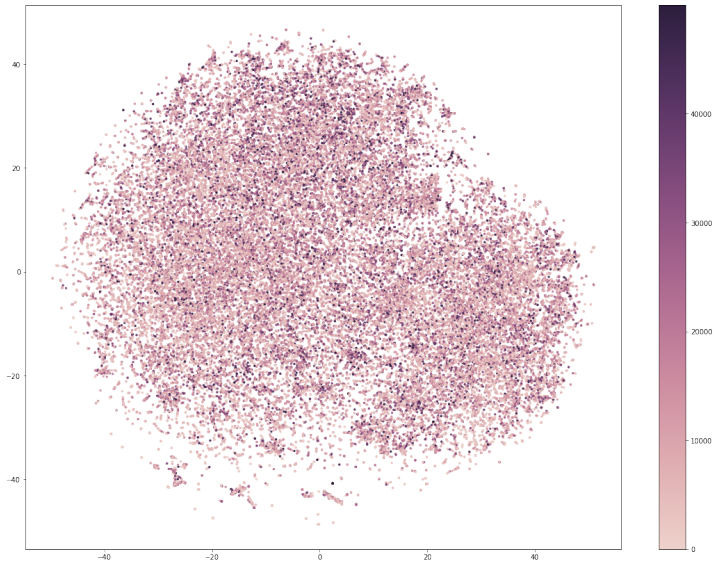


Figure 4.6: t-SNE on node2vec embeddings with the Revenue of these nodes.

4.3.2. Results

Next, we will experiment with improving existing Revenue forecasting techniques using node features as discussed in section 4.2. The results of this experiment can be found in table 4.1. In this table, we can see that the best performing model in terms of both metrics, indicated in red, is the model where we enrich the baseline model using previous degree and clustering coefficient data. The degree model seems to improve on the baseline mostly in terms of RMSE, whereas the clustering model seems to improve on the baseline in terms of MAE. The degree + clustering model combines these factors, beating the baseline model in all metrics. The node2vec approach has a very significant negative impact on the performance of the linear regression model. This might be due to the fact that a linear regression is unable to meaningfully capture the highly nonlinear

node2vec embeddings. We also see that in every model where node2vec embeddings are provided, the performance is significantly worse than the performance of the baseline, leading us to believe that of the node features we tried out, the degree and clustering coefficient of the node are the only suitable node features to improve Revenue predictions using a linear regression.

	RMSE	**MAE**
Baseline	6000.91	3768.02
Degree	5984.54	3767.42
Clustering	6000.30	3762.61
node2vec	6145.57	4076.53
Degree + clustering	**5983.53**	**3762.31**
Degree + node2vec	6347.60	4379.56
Clustering + node2vec	6372.57	4397.52
Degree + clustering + node2vec	6047.50	4011.06

Table 4.1: Results for Revenue prediction.

When we compare the baseline model to the best performing model, being the degree + clustering model, we can for both models determine whether the group of variables are statistically significant by trying to reject the null hypothesis that the group of variables is not statistically significant. For this, we choose an alpha value of 0.05. In our analysis, we can see that the probability of the group of variables being not statistically significant is 0.00 for both models. Since this is lower than our alpha value of 0.05, we can reject the null hypothesis and conclude that the group of variables used in both models are statistically significant.

Additionally, we can take a look at the goodness-of-fit by computing the R-squared value for both models. When comparing these values, we can see that the R-squared of the baseline model is 0.809, whereas the R-squared of the degree + clustering model is 0.811, indicating that the degree + clustering model can slightly explain the change in Revenue better than the baseline model. However, since the degree + clustering model utilizes three times as many variables as the degree model, we also must take the adjusted R-squared into account since it penalizes models based on the number of variables. The adjusted R-squared for the baseline model is still 0.809, whereas the adjusted R-squared for the degree + clustering model is 0.810, indicating that at least some of the enriched variables in the degree + clustering model are indeed contributing to the model.

Lastly, we will be looking at whether the individual features of the best enriched model, being the degree + clustering model, are significant by trying to reject the null hypothesis that the individual feature is not significant. Again, we will choose an alpha value of 0.05. From the previous Revenue values, we see that the Revenues from 11 and 10 months ago have a p-value of 0.133 and 0.171 respectively, meaning we cannot reject the null hypothesis for these features. However, the other previous Revenue values have a p-value < 0.05, meaning that we can reject the null hypothesis and conclude that these features are significant for Revenue prediction. For the degree features, we see that the degrees 12, 11, 8, 5, 4, 3, and 2 months ago have a p-value > 0.05, meaning that for these features, we cannot reject the null hypothesis. However, for the degree 10, 9, 7, 6, and 1 months ago, the p-value is smaller than 0.05, meaning that we can reject the null hypothesis for these features and conclude that they are significant. For the clustering coefficient features, the p-values for all features are larger than 0.05, meaning that we cannot reject the null hypothesis for any of the clustering coefficient features.

4.3.3. Post-analysis

Since the degree + clustering model only beats the baseline model by a slight margin, we are interested in if there is a certain subset of companies where this model more significantly outperforms the baseline model. Therefore, we consider the four sectors with a fairly large amount of nodes as discussed in section 3.4.4. The meanings of the sector codes can be found in appendix A. For each of these sectors, we train the baseline model and the degree + clustering model on the data from that sector. Then, we generate predictions for companies in this sector. The experiment design is the same as the one described in figure 4.3, the difference being that the input data originates from a single sector. The percentual change in RMSE and MAE of the degree + clustering model with regard to the baseline model for the four sectors and the full dataset can be found in figure 4.7. In this figure, we can firstly see that for all companies, the degree + clustering model slightly outperforms the baseline model in both metrics as reported in table 4.1. We can also see that for companies in sector G, the degree + clustering approach manages to lower the RMSE with regard to the baseline

approach. However, the baseline approach still outperforms the degree model in terms of MAE. For sector F, although slightly, the degree model outperforms the baseline model in all metrics. This decrease in error is better for companies of sector M, where the degree model significantly beats the baseline in all metrics, especially in terms of MAE. Since sector M is described as "Consultancy, research and other specialised business services", we think that the explanation of why node features work well for companies in this sector is because, for consultancy companies, the amount of revenue they generate is strictly related to the number of companies they do business with. When links for a consultancy company disappear, it is understandable that this likely means a decrease in Revenue. For other companies, this is not necessarily the case. It is apparent that for companies in sector K, the baseline vastly outperforms the degree + clustering model in terms of both metrics.

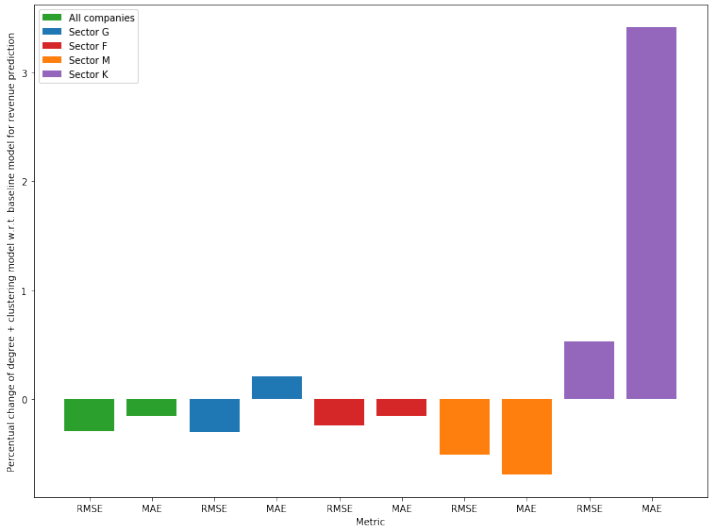


Figure 4.7: Percentual change in RMSE and MAE of the degree + clustering model with regard to the baseline model for Revenue prediction per sector.

Alternatively, we can take a look at whether the degree + clustering model outperforms the baseline model for nodes with a certain degree. To analyze this, we split up the companies into three categories: the companies with an low degree, the companies with a medium degree, and the companies with a high degree. Each of these categories consists of approximately one-third of the companies. This means that the low degree category consists of companies with an average degree lower or equal to 2.25, the medium degree category consists of companies with an average degree higher than 2.25 and lower or equal to 4.42, and the high degree category consists of companies with an average degree higher than 4.42. The results for the baseline model and the degree + clustering model for these three categories can be found in table 4.2. In this table, we can firstly see that for both models in an absolute manner, they both score better for companies with a low degree than for companies with a high degree. We think that this has to do with the fact that as shown in figure 4.5a, there is a positive correlation between degree and Revenue. Since the range of mean degrees in the high degree category is by far the largest, we think that the range in Revenues to predict is also the largest, making accurate predictions more difficult, resulting in larger errors. For companies with a low degree, both models perform similarly, with the baseline model slightly outperforming the degree + clustering model. This also holds for companies with a medium degree. However, we can see that for companies with a high degree, the degree + clustering model beats the baseline model in terms of both metrics, especially in terms of MAE, where the degree + clustering model lowers the error by ~0.8% with regard to the baseline model. We think that the degree + clustering model outperforms the baseline model especially for companies with a high degree due to the degree distribution as shown in figure 3.4. Since the degree follows a power-law distribution, there is more degree variance in the high degree category, which provides more useful information for predictions.

We are also interested in whether the degree + clustering model outperforms the baseline model for companies with certain Revenue. Similarly to the degree categories as described above, we can also categorize all companies into three approximately equally sized categories: companies with a low mean Revenue, companies with a medium mean Revenue, and companies with a high mean Revenue. Again, each of these cat-

	Low degree		**Medium degree**		**High degree**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	4788.19	2574.96	5745.37	3632.92	7725.35	5475.25
Degree + clustering	4793.73	2577.43	5751.17	3639.13	7705.10	5432.39

Table 4.2: Results for Revenue prediction for the various degree categories.

egories consists of approximately one-third of the companies. This means that the low Revenue category consists of companies with a mean Revenue up to 6886.80, the medium Revenue category consists of companies with a mean Revenue higher than 6886.80 and up to 13976.82, and the high Revenue category consists of companies with a mean Revenue higher than 13976.82. The results for the baseline model and the degree + clustering model for these three categories can be found in table 4.3. In this table, we can see that absolutely, both the baseline model and the degree model perform worse for the high Revenue category than for the other two. However, this is to be expected as we expect the high Revenue category to be a lot larger in terms of possible values to predict than the other two categories, making predictions more difficult. Also, we see that for companies in the low Revenue category, the degree + clustering model outperforms the baseline model, improving on the baseline by ~0.9% in terms of RMSE and ~1.3% in terms of MAE. However, this changes for the medium and high Revenue category, where the baseline model and the degree + clustering model perform similarly.

	Low Revenue		**Medium Revenue**		**High Revenue**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	2036.17	1762.86	1881.50	1567.53	6740.33	5045.59
Degree + clustering	2018.55	1740.10	1881.97	1567.75	6757.50	5041.20

Table 4.3: Results for Revenue prediction for the various Revenue categories.

Lastly, we will look at the performance of the baseline model versus the degree + clustering model for three different months as discussed in section 4.2. To reiterate, we will take a look at predictions for a month before any coronavirus effect, namely September 2019, a month where the coronavirus effects were noticeable for the first time, namely April 2020, and a month deeper into the COVID-19 pandemic, namely September 2020. The results of this experiment can be found in table 4.4. In this table, we can firstly see that for September 2019, both models perform the best, followed by September 2020, and lastly, April 2020. This is to be expected since historical data is the most representative for September 2019. In September 2020, we already had some of the coronavirus effects in our historical data, making the predictions more accurate than the predictions for April 2020, where the coronavirus effects were newer and more unpredictable. For September 2019, the degree + clustering model outperforms the baseline slightly. However, this difference is larger for April 2020, where the degree + clustering model lowers the RMSE by ~0.3% and the MAE with ~0.2% compared to the baseline. For September 2020, the baseline outperforms the degree + clustering model in terms of MAE, but the degree + clustering model beats the baseline model very slightly in terms of RMSE. These findings support our hypothesis that for predictions for April 2020, node features can potentially further improve KPI predictions than for other months since historical data is less representative at the start of a large change such as the COVID-19 lockdown.

	September 2019		**April 2020**		**September 2020**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	4842.90	2837.39	6000.91	3768.02	5655.15	3311.80
Degree + clustering	4832.70	2835.49	5983.53	3762.31	5653.40	3316.91

Table 4.4: Results for Revenue prediction for various months.

4.4. CashFlowMonthly forecasting

In this section, we will take a look at CashFlowMonthly forecasting and how to improve these forecasts using node features using the experiments as described in section 4.2.

4.4.1. Pre-analysis

Similarly to the analysis for Revenue as described in section 4.3, we will start by taking a look at the distribution of the KPI we want to predict. Therefore, in figure 4.8, the CashFlowMonthly distribution is depicted. In this figure, we can see that CashFlowMonthly follows a normal distribution that is centered around 0.

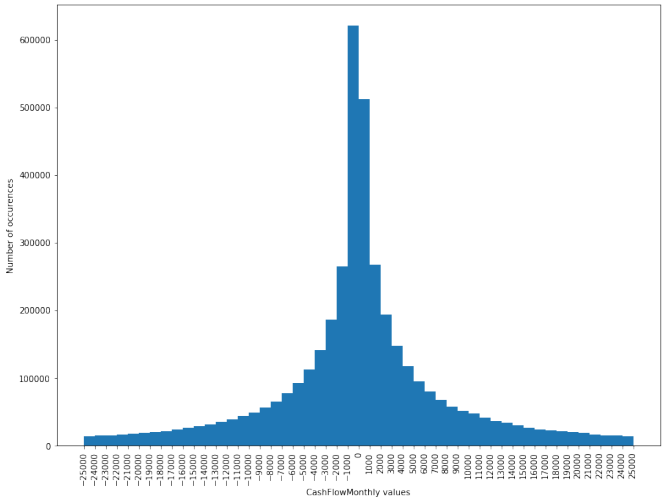
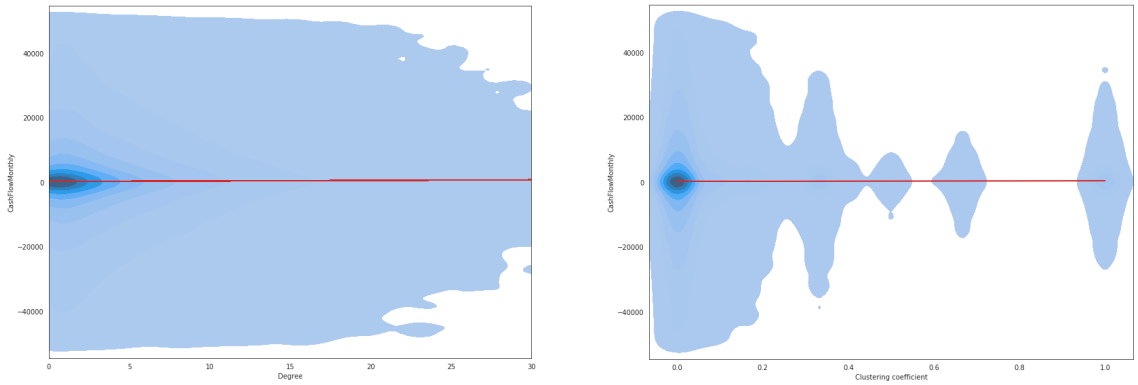


Figure 4.8: Histogram depicting the CashFlowMonthly distribution.

Next, we will take a look at the correlations between the node features we will provide and CashFlowMonthly in figure 4.9. In figure 4.9a, a kernel density plot with a linear regression line for degree and CashFlowMonthly is depicted. In this plot, we can not distinguish a clear correlation between the degree and CashFlowMonthly. The same goes for the clustering coefficient as depicted in figure 4.9b, where there is no clear correlation to be seen as well.



(a) Density plot and linear regression line indicating the correlation between degree and CashFlowMonthly.

(b) Density plot and linear regression line indicating the correlation between clustering coefficient and CashFlowMonthly.

Figure 4.9: Density plots and linear regression lines indicating the correlation between various node features and CashFlowMonthly.

Additionally, we will also be looking at the relationship between t-SNE on the node2vec embeddings and the CashFlowMonthly of these nodes as shown in figure 4.10. In this figure, the location of the point indicates the values of the t-SNE dimensionality reduction on the node2vec embedding of the node, whereas the color of the point indicates the CashFlowMonthly of the node. Unfortunately, we cannot clearly see any clusters of nodes with a similar CashFlowMonthly.

4.4.2. Results

After analyzing the potential usefulness of the various node features, we will run the experiments for forecasting CashFlowMonthly. The results of these experiments can be found in table 4.5. In this table, we can see that the best-performing model in terms of all metrics is the degree model. Enriching the baseline with

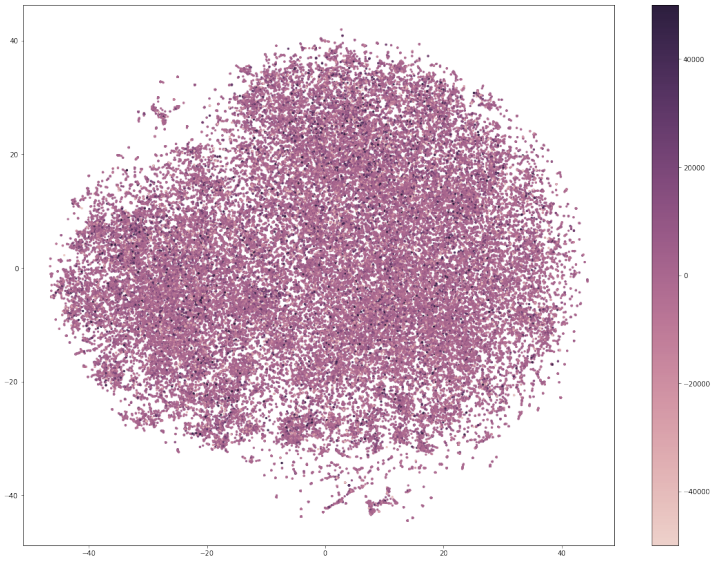


Figure 4.10: t-SNE on node2vec embeddings with the CashFlowMonthly of these nodes.

the clustering coefficient of the corresponding node does not seem to have a very significant impact on the performance of the model. Similar to Revenue prediction, adding the node2vec embeddings however seems to deteriorate performance significantly.

	RMSE	**MAE**
Baseline	9793.97	5928.87
Degree	**9782.93**	**5908.53**
Clustering	9794.18	5928.55
node2vec	9821.72	6026.57
Degree + clustering	9783.83	5909.31
Degree + node2vec	9813.80	6012.72
Clustering + node2vec	9816.01	6017.23
Degree + clustering + node2vec	9808.20	5985.54

Table 4.5: Results for CashFlowMonthly prediction.

First, we will take a look at whether the variables used by the baseline model and the degree model are statistically significant by trying to reject the null hypothesis that the variables are not statistically significant. For this, we will use an alpha value of 0.05. In our analysis we can see that the probability of the variables being insignificant is 1.93e-261 for the baseline model and 0.00 for the degree model, meaning that for both models, we can reject the null hypothesis and conclude that the variables that are used are statistically significant.

Next, we will compare the goodness-of-fit from both models by comparing the R-squared values. This is 0.018 for the baseline model and 0.025 for the degree model. However, since the R-squared value is non-decreasing when adding more features, we will also look at the adjusted R-squared values, which punish models based on the number of features it utilizes. The adjusted R-squared values are the same as the R-squared values, being 0.018 for the baseline model and 0.025 for the degree model, meaning that the degree model can explain the change in Revenue better than the degree model, and the degree features are indeed contributing to the model.

Lastly, we will try to see which individual features are significant by trying to reject the null hypothesis that a feature is insignificant. For this, we will use an alpha value of 0.05. For the features that include the previous CashFlowMonthly value, we can see that for the CashFlowMonthly 8 and 4 months ago, we have a p-value larger than 0.05, meaning that we cannot reject the null hypothesis. However, for the other features, we can reject the null hypothesis and thus conclude that the features are significant. For the degree features, we can see that for the degree 7 and 5 months ago, we have a p-value larger than 0.05 and thus cannot reject the null hypothesis for these features. For the other degree features, we can conclude that they are significant.

4.4.3. Post-analysis

Similar to the approach for Revenue prediction, we are interested in how our degree model for CashFlowMonthly predictions performs with regard to the baseline for companies in various sectors. The percentual change of the degree model with regard to the baseline model for the various sectors is shown in figure 4.11. In this figure, we see that for all companies, the degree model outperforms the baseline model in both RMSE and MAE. For companies in sector M, the degree model outperforms the baseline model even further than for all companies. Similarly to Revenue prediction, we think that the explanation of why node features work well for companies in this sector is because, for consultancy companies, the amount of cash flow they generate is strictly related to the number of companies they do business with. When links for a consultancy company disappear, it is understandable that this likely means a decrease in CashFlowMonthly. However, we can also see that for companies in sectors G, F, and K, the baseline model outperforms the degree model.

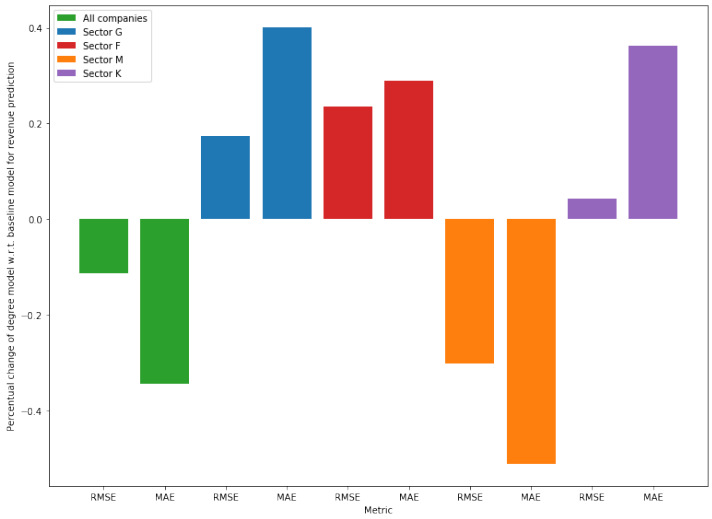


Figure 4.11: Percentual change in RMSE and MAE of the degree model with regard to the baseline model for CashFlowMonthly prediction per sector.

Next, we want to verify whether the degree model outperforms the baseline model for nodes with a certain degree. Similarly to Revenue prediction as discussed in section 4.3, we divide the nodes into three separate categories: nodes with a low degree, nodes with a medium degree, and nodes with a high degree. All three categories are approximately the same size in terms of the number of companies. Nodes with a low degree are defined as nodes with a mean degree up to 2.42, nodes with a medium degree are defined as nodes with a mean degree higher than 2.42 and up to 4.92, and nodes with a high degree are defined as nodes with a degree higher than 4.92. These boundaries are slightly different than the ones for Revenue prediction because a different set of companies has CashFlowMonthly computed for them as illustrated in figure 4.1, leading to slightly different boundary values for the degree categories. The results can be found in table 4.6. In this table, can firstly see that the predictions for high degree companies are typically worse than for companies with a low or medium degree. Next, we can see that for companies with a low average degree, the baseline model outperforms the degree model. For the medium degree companies, the baseline model still outperforms the degree model, although by a relatively lower margin than for companies with a low degree. This changes for the high degree companies, where the degree model manages to, although slightly, improve upon the baseline model. Similarly to Revenue prediction, we think that the degree model outperforms the baseline model for companies with a high degree due to the degree distribution as shown in figure 3.4. Since the degree follows a power-law distribution, there is more degree variance in the high degree category, which generally provides more useful information for predictions.

Lastly, we are interested in whether the degree model outperforms the baseline model for companies with a certain mean CashFlowMonthly. Therefore, we split up the companies into three categories: companies with a low CashFlowMonthly, companies with a medium CashFlowMonthly, and companies with a high CashFlowMonthly. Companies with a low CashFlowMonthly are defined as companies with a mean CashFlowMonthly lower or equal to -168.89. Companies with a medium CashFlowMonthly are defined as companies with a mean CashFlowMonthly higher than -168.89 and lower or equal to 520.27, whereas companies

	Low degree		**Medium degree**		**High degree**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	7844.15	4290.72	8975.29	5456.82	12398.68	8233.17
Degree	7851.20	4308.66	8978.68	5461.74	12378.54	8218.49

Table 4.6: Results for CashFlowMonthly prediction for the various degree categories.

with a high CashFlowMonthly are defined as companies with a mean CashFlowMonthly higher than 520.27. Again, these three categories are approximately the same size in terms of the number of companies per category. The results of this experiment can be found in table 4.7. In this table, we can firstly see that the error for medium CashFlowMonthly predictions is significantly lower than for the low and high CashFlowMonthly categories. We think that this has to do with the fact that the range of values in the medium CashFlowMonthly category is probably significantly smaller than for the low and high CashFlowMonthly categories, resulting in lower errors. Also, we can see that for low CashFlowMonthly, the degree model outperforms the baseline, where the degree model lowers the RMSE with regard to the baseline model by ~0.6% and the MAE by ~1.0%. However, for companies with a medium CashFlowMonthly, the baseline model and the degree model perform similarly, with the baseline model slightly outperforming the degree model. For companies with a high CashFlowMonthly, the degree model outperforms the baseline model, lowering RMSE and MAE by respectively ~0.6% and ~0.8%. It seems to be the case that for CashFlowMonthly close to 0, the degree data does not provide a significant improvement over the baseline model. However, for values that are further away from 0, adding the degree data can yield improvements over the baseline model.

	Low CashFlowMonthly		**Medium CashFlowMonthly**		**High CashFlowMonthly**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	7165.64	4905.49	184.71	157.04	7099.68	5275.08
Degree	7122.83	4855.18	186.13	157.33	7059.62	5231.10

Table 4.7: Results for CashFlowMonthly prediction for the various CashFlowMonthly categories.

Lastly, we will look at the performance of the baseline model versus the degree model for three different months. To reiterate, we will take a look at predictions for a month before any coronavirus effect, namely September 2019, a month where the coronavirus effects were noticeable for the first time, namely April 2020, and a month deeper into the COVID-19 pandemic, namely September 2020. These results can be found in figure 4.8. In this table, we can see that the predictions of both the baseline model and the degree model for September 2019 are significantly better than the predictions for April 2020 and September 2020, which is to be expected as historical data is more informative to the value to be predicted in September 2019. For predictions for September 2019, we can see that the baseline model slightly outperforms the degree model for both RMSE and MAE. However, for April 2020, the degree model outperforms the baseline model for both metrics. For September 2020, both models perform similarly, with the degree model slightly outperforming the baseline model for both metrics. These findings support our hypothesis that for predictions for April 2020, node features can potentially further improve KPI predictions than for other months since historical data is less representative at the start of a large change such as the COVID-19 lockdown.

	September 2019		**April 2020**		**September 2020**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	8705.14	5064.57	9793.97	5928.87	10031.44	6043.99
Degree	8705.40	5067.44	9782.93	5908.53	10028.38	6041.14

Table 4.8: Results for CashFlowMonthly prediction for various months.

4.5. D2C_14d forecasting

In this section, we will take a look at D2C_14d forecasting and how to improve these forecasts using node features using the experiments as described in section 4.2.

4.5.1. Pre-analysis

For D2C_14d forecasting, we will start by taking a look at the distribution of the D2C_14d values as depicted in figure 4.12. In this figure, we can see that a fairly low amount of companies get their 14-day invoices paid in a low amount of days. The amount of occurrences is increasing between 0 and 14 days, then stays roughly equal up to 22 days, and is generally decreasing afterward. However, there does not seem to be a normal distribution since the distribution does not seem to be symmetrical around the mean.

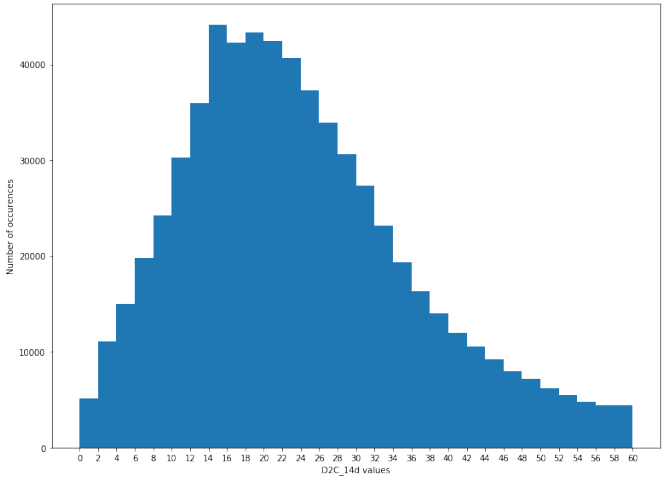
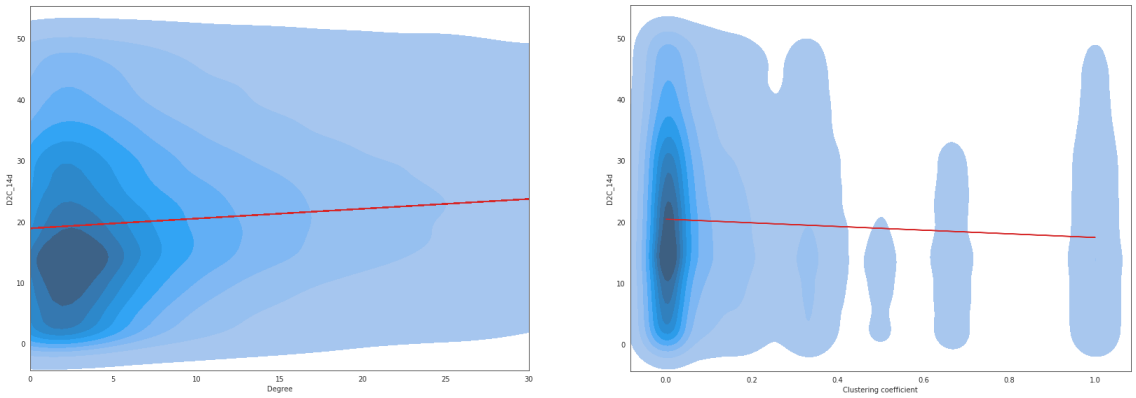


Figure 4.12: Histogram depicting the D2C_14d distribution.

Similarly to Revenue and CashFlowMonthly forecasting, we will take a look at the correlations between the various node features and the KPI we want to predict, being D2C_14d. These correlations are depicted in figure 4.13. In figure 4.13a, the correlation between the degree of a node and the value for D2C_14d can be seen. In this figure, we can see that there does seem to be a slight positive correlation, meaning that degree could potentially be an informative feature for D2C_14d forecasting. In figure 4.13b, the correlation between the clustering coefficient and D2C_14d is shown. Here, we can see a slight negative correlation between the clustering coefficient and D2C_14d.



(a) Density plot and linear regression line indicating the correlation between degree and D2C_14d.

(b) Density plot and linear regression line indicating the correlation between clustering coefficient and D2C_14d.

Figure 4.13: Density plots and linear regression lines indicating the correlation between various node features and D2C_14d.

Additionally, we will also be looking at the relationship between t-SNE on the node2vec embeddings and the D2C_14d of these nodes as shown in figure 4.14. In this figure, the location of the point indicates the values of the t-SNE dimensionality reduction on the node2vec embedding of the node, whereas the color of the point indicates the D2C_14d of the node. In this figure, we cannot clearly see clusters of points with a similar value for D2C_14d.

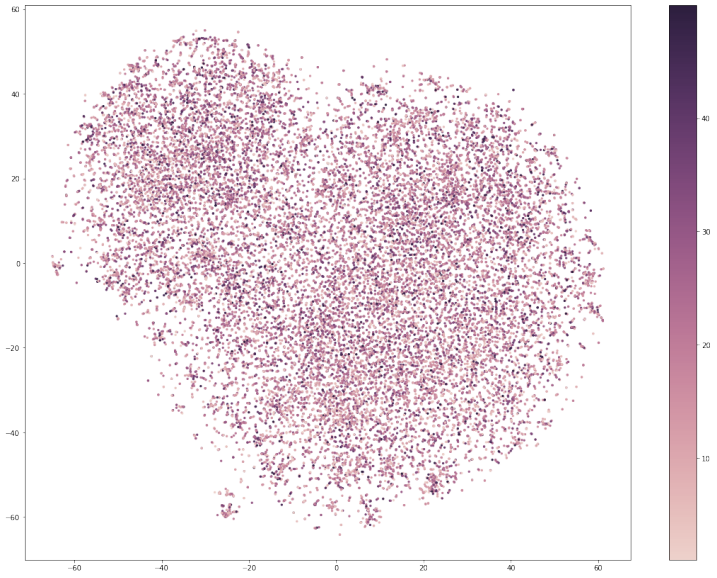


Figure 4.14: t-SNE on node2vec embeddings with the D2C_14d of these nodes.

4.5.2. Results

Now that we have analyzed the potential usefulness of the features, we will take a look at the results of the experiments. These results can be found in table 4.9. In this table, we see that both the degree model and the clustering model manage to beat the baseline model in both metrics. All models that include node2vec data once again seem to significantly deteriorate the performance with regards to the baseline model. The best-performing model is the degree + clustering model, which beats all other models in all metrics.

	RMSE	**MAE**
Baseline	7.5465	5.5142
Degree	7.5290	5.4864
Clustering	7.5397	5.5108
node2vec	7.7245	5.7542
Degree + clustering	**7.5223**	**5.4823**
Degree + node2vec	8.2948	6.2829
Clustering + node2vec	7.7361	5.7536
Degree + clustering + node2vec	8.1336	6.1358

Table 4.9: Results for D2C_14d prediction.

Firstly, we will start by checking whether the sets of features used by the baseline model and the best enriched model, being the degree + clustering model, are significant by trying to reject the null hypothesis that the sets of features are insignificant. For this, we will use an alpha value of 0.05. In our analysis we can see that the probability of the variables being insignificant is 0.00 for both models, meaning that for both of these models, we can reject the null hypothesis and conclude that the variables that are used are statistically significant.

Then, we will examine the goodness-of-fit of both models by comparing the R-squared values for the baseline model and the degree + clustering model. The R-squared value of the baseline model is 0.893, whereas the value of the R-squared value for the degree + clustering model is 0.894, indicating that the degree + clustering model is slightly better at explaining the change in D2C_14d than the baseline model. However, since the R-squared value is non-decreasing when adding more features, we also want to compare the adjusted R-squared values of the baseline and degree + clustering model, since it penalizes models with more features more heavily. We can see that for both models, the R-squared values are equal to the adjusted R-squared values, being 0.893 for the baseline model and 0.894 for the degree + clustering model, indicating that although more features are added with the degree + clustering model, the goodness-of-fit of the degree + clustering model is indeed slightly better.

Lastly, we will examine whether the individual features that are used by the degree + clustering model are statistically significant by trying to reject the null hypothesis that the individual feature is insignificant. Of the features modeling the previous D2C_14d values, the D2C_14d value from 7 months ago has a p-value larger than 0.05, meaning that this feature is the only previous D2C_14d value for which we cannot reject the null hypothesis. For the degree values 11, 9, 6, 5, and 3 months ago, we also have a p-value larger than 0.05, meaning that we cannot conclude that these features are significant. For the other degree value features, we can reject the null hypothesis and thus conclude that the features are significant. Similarly to Revenue prediction as discussed in section 4.3, we do not have a p-value smaller than 0.05 for any of the clustering coefficient features, meaning that we cannot reject the null hypothesis for any of these features.

4.5.3. Post-analysis

Similarly to Revenue and CashFlowMonthly forecasting, we will start by examining the change in RMSE and MAE of the best model, being the degree + clustering model, with regards to the baseline model per sector. The results of this experiment can be found in figure 4.15. In this figure, we can see that for all companies, the degree + clustering model slightly outperforms the baseline model. We can also see that, similarly, for Revenue and CashFlowMonthly forecasting, this improvement is larger for companies from sector M. It also stands out that for companies in sector K, the degree + clustering model performs significantly worse than the baseline model, especially in terms of MAE.

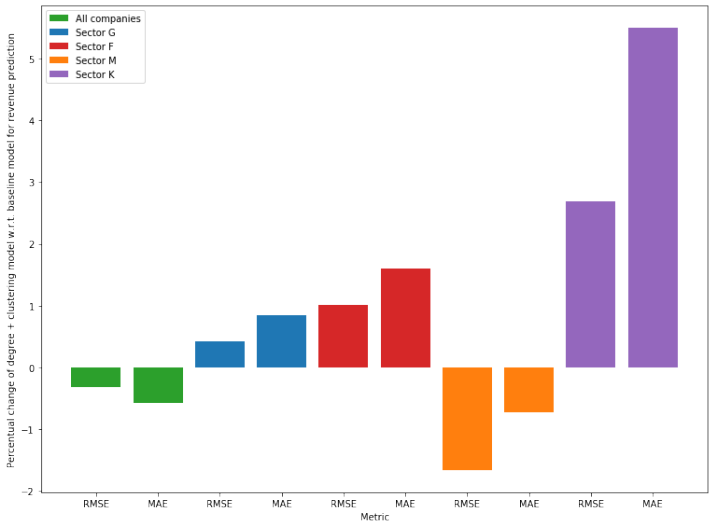


Figure 4.15: Percentual change in RMSE and MAE of the degree + clustering model with regard to the baseline model for D2C_14d prediction per sector.

Alternatively, we will take a look at if the degree + clustering model outperforms the baseline model for companies with a certain degree. Similarly to Revenue prediction as discussed in section 4.3 and CashFlowMonthly description as discussed in section 4.4, we divide the nodes into three separate categories: nodes with a low degree, nodes with a medium degree, and nodes with a high degree. All three categories are approximately the same size with regard to the number of companies in them. Nodes with a low degree are defined as nodes with a mean degree up to 5.00, nodes with a medium degree are defined as nodes with a mean degree higher than 5.00 and up to 12.58, and nodes with a high degree are defined as nodes with a degree higher than 12.58. These boundaries are different than the ones for Revenue and CashFlowMonthly prediction because a different set of companies has D2C_14d computed for them as illustrated in figure 4.1, leading to different boundary values for the degree categories. These results can be found in table 4.10. For companies with a low degree and companies with a medium degree, we can see that the baseline model manages to outperform the degree + clustering model for both metrics. However, similarly to Revenue and CashFlowMonthly forecasting, for companies with a high degree, the degree + clustering model outperforms the baseline model, lowering the RMSE and MAE by respectively ~1.2% and ~1.0%. We think that the degree + clustering model outperforms the baseline model especially for companies with a high degree due to the degree distribution as shown in figure 3.4. Since the degree follows a power-law distribution, there is more degree variance in the high degree category, which provides more useful information for predictions.

	Low degree		**Medium degree**		**High degree**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	7.6657	5.3981	7.6351	5.5884	7.4107	5.5636
Degree + clustering	7.6885	5.4249	7.6474	5.6121	7.3211	5.5098

Table 4.10: Results for D2C_14d prediction for the various degree categories.

Additionally, we are interested in whether the degree + clustering model outperforms the baseline model for companies with a certain mean D2C_14d. Therefore, we split up the companies into three categories: companies with a low D2C_14d, companies with a medium D2C_14d, and companies with a high D2C_14d. Companies with a low D2C_14d are defined as companies with a mean D2C_14d lower or equal to 16.94. Companies with a medium D2C_14d are defined as companies with a mean D2C_14d higher than 16.94 and lower or equal to 23.46, whereas companies with a high D2C_14d are defined as companies with a mean D2C_14d higher than 23.46. Again, these three categories are approximately the same size in terms of the number of companies per category. The results of this experiment can be found in table 4.11. In this table, we can see that for low D2C_14d, the degree + clustering model outperforms the baseline model. However, for companies with a medium and high D2C_14d to predict, the baseline model outperforms the degree + clustering model.

	Low D2C_14d		**Medium D2C_14d**		**High D2C_14d**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	3.589	2.8787	1.8526	1.6199	5.8195	4.6950
Degree + clustering	3.5801	2.8665	1.8678	1.6347	5.8471	4.7187

Table 4.11: Results for D2C_14d prediction for the various D2C_14d categories.

Lastly, we will look at the performance of the baseline model versus the degree + clustering model for three different months. To reiterate, we will take a look at predictions for a month before any coronavirus effect, namely September 2019, a month where the coronavirus effects were noticeable for the first time, namely April 2020, and a month deeper into the COVID-19 pandemic, namely September 2020. These results can be found in figure 4.12. In this table, we see that for both the baseline and the degree + clustering model, D2C_14d predictions for April 2020 are worse than the predictions for September 2019 and September 2020, which is in line with our expectations. For predictions for September 2019, the baseline and the degree + clustering model perform similarly, with the baseline model outperforming the degree + clustering model in terms of MAE, but the degree + clustering model outperforming the baseline model in terms of RMSE. However, for predictions for April 2020, the degree + clustering model outperforms the baseline model, lowering the RMSE by ~0.3% and the MAE by ~0.6%. For predictions for September 2020, the baseline model and the degree + clustering model again perform similarly, with the baseline model outperforming the degree + clustering model in terms of MAE, and the degree + clustering model outperforming the baseline model in terms of RMSE. These findings support our hypothesis that for predictions for April 2020, node features can potentially further improve KPI predictions than for other months since historical data is less representative at the start of a large change such as the COVID-19 lockdown.

	September 2019		**April 2020**		**September 2020**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Baseline	7.5064	5.4099	7.5465	5.5142	6.9159	4.8977
Degree + clustering	7.5061	5.4127	7.5223	5.4823	6.9134	4.9048

Table 4.12: Results for D2C_14d prediction for various months.

5

Conclusion and future work

In this chapter, we will recap the findings that were discussed in this thesis to answer the research question as defined in section 1.1. Additionally, we will discuss some of the future work that could be done to improve upon the work presented in this thesis.

5.1. Conclusion

In section 1.1, we defined the research question as: *Can we utilize a dynamic network of SMEs to improve KPI predictions during the COVID-19 lockdown?* To answer this question, we split up the research question into two separate subquestions for us to look at. In this section, we will discuss both subquestions and the steps we took to answer them, to ultimately answer our research question.

The first subquestion was defined as: *How can we create a dynamic network of SMEs out of unstandardized data?* To answer this question, we developed a novel entity resolution algorithm to find a mapping between companies. This entity resolution first performs a cleaning step to filter out accounts that do not have reliable information available. Then, we split the accounts based on the type and try to find a matching between them using three different types of matchings; Chamber of Commerce number matching, where we match accounts based on their Chamber of Commerce number, VAT number matching, where we match accounts based on their VAT number, and fuzzy matching, where we match accounts based on their ZIP code, email address, and the Levenshtein similarity between the company names. Using these matched accounts, we use transactions from January 1st, 2018 to September 30th, 2020 to build our dynamic network. This dynamic network consists of a series of static networks, one for each month in the period between January 2018 and September 2020, where an edge is present between two nodes if the respective companies had at least one transaction between them in the corresponding month. When analyzing this network, we can see, for example, the impact of the coronavirus lockdown in several features, such as the number of active nodes, the number of edges, the average degree, and the standard deviation of degree.

The second subquestion was defined as: *How can we utilize the dynamic network to improve KPI predictions?* Next to seeing the influence of the COVID-19 lockdown on Dutch SMEs in the dynamic network we created, to assess the influence of the COVID-19 lockdown on Dutch SMEs, we take a look at the relative change with regard to the same month last year for several Key Performance Indicators or KPIs for short. For some KPIs, being Revenue, CashFlowMonthly, and the various D2C KPIs, we can see a clear impact of the COVID-19 lockdown on Dutch SMEs. In the next step, we try to improve upon the forecasting of KPIs where we can see the impact of the coronavirus lockdown, being Revenue, CashFlowMonthly, and one of the D2C KPIs, namely D2C_14d. We try to improve forecasting of KPI predictions for April 2020, since the coronavirus lockdown effects are the largest in this month. We hypothesize that for predictions for April 2020, node features can potentially further improve KPI predictions than for other months since historical data is less representative at the start of a large change such as the COVID-19 lockdown. For Revenue forecasting, we found that enriching the historical data with the degree and clustering coefficient of the corresponding node in the dynamic network can improve Revenue predictions. However, none of the clustering coefficient features were statistically significant for Revenue forecasting. We also found that these predictions are especially better for companies in sector M, companies with a high average degree in the network, and companies with a relatively low average Revenue. For CashFlowMonthly forecasting, we found that enriching the historical

data with the degree of the corresponding nodes decreases the error of the predictions. Similar to Revenue prediction, companies in sector M and companies with a relatively high average degree benefit more from the enrichment with node features. This also holds for companies with a relatively low or high average CashFlowMonthly. For D2C_14d forecasting, we found that the best performing model was the one where we enrich the historical data with the degree and the clustering coefficient of the corresponding node. However, similarly to Revenue prediction, none of the clustering coefficient features were statistically significant for D2C_14d forecasting. Similarly to Revenue and CashFlowMonthly prediction, companies from sector M and companies with a relatively high average degree benefit more from the enrichment with node features. This also holds for companies with a relatively low average D2C_14d. We also verified our hypothesis that for all KPIs, predictions for April 2020 benefit more from the enrichment with node features than other months.

To finally answer the research question of whether we can utilize a dynamic network of SMEs to improve KPI predictions during the COVID-19 lockdown, we first build a dynamic network consisting of Dutch SMEs and the transactions between them. In this dynamic network, we can see the effects of the coronavirus lockdown on Dutch SMEs in some properties of the network. These effects are also reflected in some of the KPIs of the companies in the network. Using information contained in the network, we can improve upon KPI forecasting techniques at the beginning of the COVID-19 lockdown by enriching historical data with node features.

5.2. Future work

In this project, we have developed the basic framework of methods to address the general question of how to improve KPI prediction using the dynamic network of SMEs. We deem the following directions as promising future work.

5.2.1. Entity resolution

In section 3.2 we discussed the lightweight entity resolution algorithm that was used to build the dynamic network. As previously discussed, an entity resolution algorithm was provided in [14] but was infeasible to run on the dataset due to the computational complexity of the algorithm. A more elaborate but computational efficient algorithm is desirable. Moreover, if one would have significantly more time to run an entity resolution algorithm, one could run the entity resolution algorithm as presented in [14] to compare the results against the entity resolution algorithm as described in this thesis. Additionally, one could expand the algorithm presented in 3.2 with more matching fields such as the IBANs of the companies.

5.2.2. KPI forecasting

In section 4.3 we discussed improving KPI forecasting using node features. In this section, a random forest regression instead of a linear regression was also tried out, but the random forest regressions were outperformed by their respective linear regressions in terms of both RMSE and MAE. In future work, one could explore different regression models other than the linear regression model and the random forest regression model to verify whether a performance improvement can be realized. Additionally, other node features could be considered for the improvement of KPI forecasting, such as the PageRank, as described in [28], or a dynamic node embedding, as discussed in section 2.3, of a node.

Next to further improving the KPI forecasting techniques, more work could be done to examine why an approach enriched with node features works better for some companies. For example, it is still not fully clear why predictions using node features for companies in sector M are relatively better than predictions for companies in sector K, and why predictions for companies that have a relatively low Revenue and D2C_14d were often better when using node features.

5.2.3. Link prediction

The dynamic network of Dutch SMEs constructed out of unstandardized data would allow the exploration of the link prediction problem, i.e. predicting which companies might start doing business with each other. Several techniques for link predictions have been discussed in section 2.4.

A

ISIC section codes

ISIC sections have been defined by the United Nations and are used to classify companies into several categories[1] indicating economical activities. For each section, a one-letter code is available. For every code, the corresponding description can be found below:

- A: Agriculture, forestry and fishing
- B: Mining and quarrying
- C: Manufacturing
- D: Electricity, gas, steam and air conditioning supply
- E: Water supply; sewerage, waste management and remediation activities
- F: Construction
- G: Wholesale and retail trade; repair of motor vehicles and motorcycles
- H: Transportation and storage
- I: Accommodation and food service activities
- J: Information and communication
- K: Financial institutions
- L: Renting, buying and selling of real estate
- M: Consultancy, research and other specialised business services
- N: Renting and leasing of tangible goods and other business support services
- O: Public administration, public services and compulsory social security
- P: Education
- Q: Human health and social work activities
- R: Culture, sports and recreation
- S: Other service activities
- T: Activities of households as employers; undifferentiated goods- and service- producing activities of households for own use
- U: Extraterritorial organisations and bodies

[1] `https://unstats.un.org/unsd/publication/seriesm/seriesm_4rev4e.pdf`, page 58

Bibliography

[1] Louis Abraham. fastnode2vec, 2020. URL `https://doi.org/10.5281/zenodo.3902632`.

[2] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[3] Ibrahim Said Ahmad, Azuraliza Abu Bakar, Mohd Ridzwan Yaakub, and Shamsuddeen Hassan Muhammad. A survey on machine learning techniques in movie revenue prediction. *SN Computer Science*, 1 (4):1–14, 2020.

[4] Moran Beladev, Lior Rokach, Gilad Katz, Ido Guy, and Kira Radinsky. tdgraphembed: Temporal dynamic graph-level embedding. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 55–64, 2020.

[5] Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Netsimile: A scalable approach to size-independent network similarity. *arXiv preprint arXiv:1209.2684*, 2012.

[6] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[7] Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific reports*, 3(1):1–14, 2013.

[8] Laura C Carpi, Osvaldo A Rosso, Patricia M Saco, and Martín Gómez Ravetti. Analyzing complex networks evolution through information theory quantifiers. *Physics Letters A*, 375(4):801–804, 2011.

[9] Jinyin Chen, Jian Zhang, Xuanheng Xu, Chenbo Fu, Dan Zhang, Qingpeng Zhang, and Qi Xuan. E-lstm-d: A deep learning framework for dynamic network link prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[10] Pierre-Olivier Gourinchas, Sebnem Kalemli-Özcan, Veronika Penciakova, and Nick Sander. Covid-19 and sme failures. Technical report, National Bureau of Economic Research, 2020.

[11] Robert Gove. Gragnostics: Fast, interpretable features for comparing graphs. In *2019 23rd International Conference Information Visualisation (IV)*, pages 201–209. IEEE, 2019.

[12] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

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

[14] Arthur Hovanesyan. Late payment prediction of invoices through graph features. 2019.

[15] Jill Juergensen, José Guimón, and Rajneesh Narula. European smes amidst the covid-19 crisis: assessing impact and policy responses. *Journal of Industrial and Business Economics*, 47(3):499–510, 2020.

[16] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 162–170. SIAM, 2013.

[17] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

[18] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[19] Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 388–396. IEEE, 2019.

[20] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.

[21] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[22] Jinsong Li, Jianhua Peng, Shuxin Liu, Lintianran Weng, and Cong Li. Tsam: Temporal link prediction in directed networks based on self-attention mechanism. *arXiv preprint arXiv:2008.10021*, 2020.

[23] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[24] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3762–3765. IEEE, 2018.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[27] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2): 025102, 2001.

[28] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[30] Lorena Poenaru-Olaru. Credit scoring prediction using graph features. 2020.

[31] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

[32] Polina Rozenshtein and Aristides Gionis. Mining temporal networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3225–3226, 2019.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[34] Mattia Tantardini, Francesca Ieva, Lucia Tajoli, and Carlo Piccardi. Comparing methods for comparing networks. *Scientific reports*, 9(1):1–19, 2019.

[35] Srishti Tomar and Ryan Compton. Vec2struc: A method towards explainable structural-based node embeddings. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (1)*, 2020.

[36] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[37] Dariusz Wójcik and Stefanos Ioannou. Covid-19 and finance: Market developments so far and potential impacts on the financial sector and centres. *Tijdschrift voor economische en sociale geografie*, 111(3): 387–400, 2020.

[38] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[39] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.