Classification, Prediction and Evaluation of Graph Neural Networks on Online Social Media Platforms

Seyed Ali Alhosseini Almodarresi Yasin

Publikationsbasierte Universitätsdissertation
zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

am Fachgebiet Internet-Technologien und -Systeme
des Hasso-Plattner-Institut

eingereicht an der
Digital-Engineering-Fakultät
der Universität Potsdam

Datum der Disputation: März, 2023

Betreuer

Prof. Dr. Christoph Meinel
Hasso Plattner Institute, University of Potsdam

Gutachter

Prof. Dr. Jürgen Pfeffer
Technical University of Munich

Prof. Dr. Aleksandar Bojchevski
University of Cologne

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-62642
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-626421

However much science thou mayest acquire
Thou art ignorant when there is no practice in thee.
Sa'di

For my Mum and Dad

Abstract

The vast amount of data generated on social media platforms have made them a valuable source of information for businesses, governments and researchers. Social media data can provide insights into user behavior, preferences, and opinions. In this work, we address two important challenges in social media analytics. Predicting user engagement with online content has become a critical task for content creators to increase user engagement and reach larger audiences. Traditional user engagement prediction approaches rely solely on features derived from the user and content. However, a new class of deep learning methods based on graphs captures not only the content features but also the graph structure of social media networks.

This thesis proposes a novel Graph Neural Network (GNN) approach to predict user interaction with tweets. The proposed approach combines the features of users, tweets and their engagement graphs. The tweet text features are extracted using pre-trained embeddings from language models, and a GNN layer is used to embed the user in a vector space. The GNN model then combines the features and graph structure to predict user engagement. The proposed approach achieves an accuracy value of 94.22% in classifying user interactions, including likes, retweets, replies, and quotes.

Another major challenge in social media analysis is detecting and classifying social bot accounts. Social bots are automated accounts used to manipulate public opinion by spreading misinformation or generating fake interactions. Detecting social bots is critical to prevent their negative impact on public opinion and trust in social media. In this thesis, we classify social bots on Twitter by applying Graph Neural Networks. The proposed approach uses a combination of both the features of a node and an aggregation of the features of a node's neighborhood to classify social bot accounts. Our final results indicate a 6% improvement in the area under the curve score in the final predictions through the utilization of GNN.

Overall, our work highlights the importance of social media data and the potential of new methods such as GNNs to predict user engagement and

detect social bots. These methods have important implications for improving the quality and reliability of information on social media platforms and mitigating the negative impact of social bots on public opinion and discourse.

Zusammenfassung

Die riesige Menge an Daten, die auf Social-Media-Plattformen generiert wird, hat sie zu einer wertvollen Informationsquelle für Unternehmen, Regierungen und Forscher gemacht. Daten aus sozialen Medien können Einblicke in das Verhalten, die Vorlieben und die Meinungen der Nutzer geben. In dieser Arbeit befassen wir uns mit zwei wichtigen Herausforderungen im Bereich der Social-Media-Analytik. Die Vorhersage des Nutzerinteresses an Online-Inhalten ist zu einer wichtigen Aufgabe für die Ersteller von Inhalten geworden, um das Nutzerengagement zu steigern und ein größeres Publikum zu erreichen. Herkömmliche Ansätze zur Vorhersage des Nutzerengagements stützen sich ausschließlich auf Merkmale, die aus dem Nutzer und dem Inhalt abgeleitet werden. Eine neue Klasse von Deep-Learning-Methoden, die auf Graphen basieren, erfasst jedoch nicht nur die Inhaltsmerkmale, sondern auch die Graphenstruktur von Social-Media-Netzwerken.

In dieser Arbeit wird ein neuartiger Graph Neural Network (GNN)-Ansatz zur Vorhersage der Nutzerinteraktion mit Tweets vorgeschlagen. Der vorgeschlagene Ansatz kombiniert die Merkmale von Nutzern, Tweets und deren Engagement-Graphen. Die Textmerkmale der Tweets werden mit Hilfe von vortrainierten Einbettungen aus Sprachmodellen extrahiert, und eine GNN-Schicht wird zur Einbettung des Nutzers in einen Vektorraum verwendet. Das GNN-Modell kombiniert dann die Merkmale und die Graphenstruktur, um das Nutzerengagement vorherzusagen. Der vorgeschlagene Ansatz erreicht eine Genauigkeit von 94,22% bei der Klassifizierung von Benutzerinteraktionen, einschließlich Likes, Retweets, Antworten und Zitaten.

Eine weitere große Herausforderung bei der Analyse sozialer Medien ist die Erkennung und Klassifizierung von Social-Bot-Konten. Social Bots sind automatisierte Konten, die dazu dienen, die öffentliche Meinung zu manipulieren, indem sie Fehlinformationen verbreiten oder gefälschte Interaktionen erzeugen. Die Erkennung von Social Bots ist entscheidend, um ihre negativen Auswirkungen auf die öffentliche Meinung und das Vertrauen in soziale Medien zu verhindern. In dieser Arbeit klassifizieren wir Social Bots auf Twitter

mit Hilfe von Graph Neural Networks. Der vorgeschlagene Ansatz verwendet eine Kombination aus den Merkmalen eines Knotens und einer Aggregation der Merkmale der Nachbarschaft eines Knotens, um Social-Bot-Konten zu klassifizieren. Unsere Endergebnisse zeigen eine 6%ige Verbesserung der Fläche unter der Kurve bei den endgültigen Vorhersagen durch die Verwendung von GNN.

Insgesamt unterstreicht unsere Arbeit die Bedeutung von Social-Media-Daten und das Potenzial neuer Methoden wie GNNs zur Vorhersage des Nutzer-Engagements und zur Erkennung von Social Bots. Diese Methoden haben wichtige Auswirkungen auf die Verbesserung der Qualität und Zuverlässigkeit von Informationen auf Social-Media-Plattformen und die Abschwächung der negativen Auswirkungen von Social Bots auf die öffentliche Meinung und den Diskurs.

Acknowledgments

With this PhD thesis, the exciting journey of my doctoral studies ends. This chapter of my life would not have been possible without the great support of many great people, whom I would like to expressly acknowledge here.

First of all, I would like to thank my family for their love and support, which was always present even from far away. I am grateful to my wonderful mother, Mahboubeh, for teaching and showing to be passionate and understanding towards everyone. I would like thank my father Dr. Seyed Mohammad Taghi Almodarresi to whom I'm always grateful as he taught me how to program, code and do math.

I would like to express my sincere gratitude to my doctoral supervisor Prof. Dr. Chritoph Meinel for giving me the opportunity to work on this topic and for the his trust and support during the last years. I am truly grateful to my second supervisor Prof. Dr. Felix Naumann for his invaluable support and guidance throughout my research. His encouraging and in-depth meetings have been instrumental in shaping my work.

I am for ever in dept for all the friends at HPI that helped me during the easy and difficult times. I would like to specially thank Raad Bin'Taref, Mina Rezaie, Pejman Najafi and Alexander Mühle for their feedback, motivation and support.

Contents

List of Tables

List of Figures

1 Introduction

This chapter provides an overall view of the research conducted in this thesis. Specifically, in Section 1.1 we describe the background contexts in which the work in this thesis was conducted and introduces the main motivations for the general research domain. Section 1.2 outlines the chapters in this thesis. Finally, in Section 1.3 we give a summary of the main research contributions in the thesis.

1.1 Research Background and Motivation

In recent years, several convolutional neural network architectures have been proposed for learning over graphs. Graph Neural Networks (GNNs) have attracted attention due to their ability to combine both node features and graph structure. This makes them ideal for dealing with data that contains network structures, such as social media data where user networks are present. GNNs have been applied to a variety of tasks including node classification, link prediction, and graph classification, among others. In social media analysis, GNNs have been used for various applications such as user profiling, recommender systems, and spam bot detection. The ability of GNNs to learn from both node features and graph structure has made them effective for modeling the complex relationships and interactions that exist in social networks. By using GNNs, researchers have been able to capture not only the content features but also the structural features of social media data. This has led to the development of more accurate and efficient models for user engagement prediction, social bot detection, and other social media analysis tasks.

This thesis investigates the applications of graph neural networks on the social media data. We first introduce the challenges in data collection on social media platforms. More over we take a look at Twitter as a social media website to retrieve tweets from it's API. We create a system that can collect the most tweets possible from the Twitter API. An important

problem in social media mining is predicting user engagement, as it plays a critical role in understanding user behavior and preferences. Traditional machine learning models have been applied to predict user engagement by relying solely on user and content features. However, these models ignore the underlying network structure of users and their interactions on social media. Graph Neural Networks (GNNs) have been shown to effectively model graph-structured data, integrating both node features and graph structure. Since social media data is inherently graph structured, it is useful to apply GNNs to predict user engagement. In this work, we explore the use of GNNs to predict user engagement on Twitter using both user and tweet features, as well as the network structure of interactions between users and tweets.

Another important task in social media mining is spam bot detection, as malicious accounts and social bots can harm users and spread misinformation. Traditional approaches to spam bot detection rely on features such as account creation date, account activity, and content features. However, these approaches do not effectively capture the complex relationships and interactions between users and their network structure. GNNs provide a powerful framework for modeling these interactions and relationships, and thus have emerged as a promising approach for spam bot detection. In this work, we explore the use of GNNs for spam bot detection on Twitter, leveraging features such as user and content features, as well as the network structure of interactions between accounts. We demonstrate the effectiveness of our proposed method on a well-known bot detection dataset and propose future extensions for more effective and efficient bot detection.

1.2 Thesis Outline and Structure

The thesis is structured in the following chapters:

In Chapter 2, we introduce the basics of Graph Neural Networks (GNNs) and the mathematical concepts necessary to understand them. We described how GNNs work with graph-structured data, combining not only the node features but also the graph structure as well. We then discuss the training process for GNNs, including the use of message-passing algorithms. To illustrate these concepts, we provided an example in the Karate Club network showing how GNNs can be used for node classification tasks. Overall, this

chapter laid the foundation for understanding the role of GNNs in predicting user engagement and classifying social bot accounts on Twitter in subsequent chapters.

In Chapter 3, we discuss the importance of collecting data from social media platforms for research purposes, particularly Twitter, which is a rich source of user-generated content. However, Twitter's API has several limitations, such as restrictions for full data and the inability to retrieve tweets older than a week. To address these limitations, we propose a system that can retrieve more live tweets than the streaming API offers by predicting their IDs based on the Snowflakes algorithm. We describe the architecture of our proposed system, which enables the collection of more tweets using only ten tokens. By implementing our system, researchers can access otherwise inaccessible data, enabling more in-depth analysis of Twitter trends and historical events.

In Chapters 4 and 5, we focus on the task of predicting user engagement on Twitter. We use an official dataset from the ACM RecSys challenge, which contains over two weeks of tweets and four types of interactions: like, retweet, reply, and quote. We train two models to predict the type of engagement: gradient boosting trees and graph neural networks.

In Chapter 6, we focus on the challenges of detecting spam bots on social media platforms, especially Twitter. We explore a new approach that uses Graph Convolutional Networks to classify accounts based on the graph structure and relationships of Twitter accounts. Our method aggregates feature information from the neighborhood of each account to improve classification accuracy. To demonstrate the effectiveness of our proposal, we tested it on a well-known bot detection dataset. Our results show that our method outperforms existing approaches for detecting social bots on Twitter. A possible future extension of this work is to deploy this method in real-time on Twitter's streaming API to detect spambots as they are created. This would enable early detection and mitigation of social bot attacks and improve the security and integrity of online discourse.

Finally, in Chapter 7, we conclude this thesis and present our final remarks on future work.

1.3 Main Contributions

The structure of this thesis involves a synthesis of various publications. Each upcoming chapter is intricately tied to a specific publication, with modifications and enhancements made to ensure that they align with the overarching objectives of the thesis. The concepts and discoveries presented in different sections of this thesis are as follows:

- **Chapter 3:** Seyed Ali Alhosseini and Christoph Meinel. **The More The Merrier: Reconstruction of Twitter Firehose**. In: *International Conference on Information Networking, ICOIN 2023, Bangkok, Thailand, January 11-14, 2023*. IEEE, 2023, 200–205. DOI: 10.1109/ICOIN56518.2023.10048898. URL: https://doi.org/10.1109/ICOIN56518.2023.10048898
- **Chapter 4:** Seyed Ali Alhosseini, Raad Bin Tareaf, and Christoph Meinel. **Engaging with Tweets: The Missing Dataset On Social Media**. In: *Proceedings of the Recommender Systems Challenge 2020*. RecSysChallenge '20. Virtual Event, Brazil: Association for Computing Machinery, 2020, 34–37. ISBN: 9781450388351. DOI: 10.1145/3415959.3415999. URL: https://doi.org/10.1145/3415959.3415999
- **Chapter 5:** Seyed Ali Alhosseini and Christoph Meinel. **Predicting User Engagements Using Graph Neural Networks on Online Social Networks**. In: *IThe 8th IEEE International Conference on Data Science and Systems, DSS 2022, Chengdu, China, December 18-21, 2022*. IEEE, 2022, 200–205
- **Chapter 6:** Seyed Ali Alhosseini, Raad Bin Tareaf, Pejman Najafi, and Christoph Meinel. **Detect Me If You Can: Spam Bot Detection Using Inductive Representation Learning**. In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. San Francisco, USA: Association for Computing Machinery, 2019, 148–153. ISBN: 9781450366755. DOI: 10.1145/3308560.3316504. URL: https://doi.org/10.1145/3308560.3316504

The following list contains paper and journal publications that were carried out during the doctoral studies but were not included in this work because the topics are not related to the thesis:

- Raad Bin Tareaf, Seyed Ali Alhosseini, and Christoph Meinel. **Does Personality Evolve? A Ten-Years Longitudinal Study from Social Media Platforms**. In: *IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, ISPA/BDCloud/SocialCom/SustainCom 2020, Exeter, United Kingdom, December 17-19, 2020*. Ed. by Jia Hu, Geyong Min, Nektarios Georgalas, Zhiwei Zhao, Fei Hao, and Wang Miao. IEEE, 2020, 1205–1213. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00179. URL: https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00179

  In this study, as second author, Ali Alhosseini conducted an extensive review of the literature, synthesizing existing research to inform the theoretical foundations of the study,

- Raad Bin Tareaf, Seyed Ali Alhosseini, Philipp Berger, Patrick Hennig, and Christoph Meinel. **Towards Automatic Personality Prediction Using Facebook Likes Metadata**. In: *14th IEEE International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2019, Dalian, China, November 14-16, 2019*. Ed. by Li Zou, Lingling Fang, Bo Fu, and Panpan Niu. IEEE, 2019, 714–719. DOI: 10.1109/ISKE47853.2019.9170375. URL: https://doi.org/10.1109/ISKE47853.2019.9170375

  In this study, while both authors were involved in data collection, Ali Alhosseini took the lead in data interpretation, drawing connections between findings and the broader research context.

- Raad Bin Tareaf, Seyed Ali Alhosseini, and Christoph Meinel. **Facial-Based Personality Prediction Models for Estimating Individuals Private Traits**. In: *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, ISPA/BDCloud/SocialCom/SustainCom 2019, Xiamen, China, December 16-18, 2019*. IEEE, 2019, 1586–1594. DOI: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00233. URL: https://doi.org/10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00233

In this study, as second author Seyed Ali Alhosseini provided expertise in statistical analysis and methodology, playing a key role in the interpretation of results.

2 Graph Convolution Neural Networks

In this chapter we introduce Graph Convolutional Neural Networks (GCNN). We will go through an example of training a simple GCNN on the Zachary's karate club network.

2.1 Graph Convolutional Neural Networks

Graphs are abstract mathematical formulations that can express many of the phenomes we see happening in around us. For example, in chemistry, molecules are graphs of atoms and the connections between them. In traffic systems, cities and roads connecting them can be represented using graphs. Social networks, which is the focus of this thesis is a clear example of the graph of users and friendship connection between them.

A graph $G = (V; E)$ consists of a set of nodes V and a set of edges E that connect these nodes. Edges are represented as $(u; v)$ in E, where u and v are nodes in V. Graphs are often represented using an adjacency matrix A, where nodes are ordered to correspond to rows and columns of the matrix. The presence of edges is indicated by entries in this matrix, where $A[u; v] = 1$ if an edge exists between nodes u and v, and $A[u; v] = 0$ otherwise.

The development of deep learning models for graph-structured data presents unique challenges. Conventional deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are designed for grid-structured inputs (e.g., images) and sequences (e.g., text), respectively. To create deep neural networks for general graphs, novel architectures must be developed.

Graph neural networks (GNNs) employ message-passing to update hidden embeddings h_u^k for each node u in V during each iteration. The update for each node u is based on information from its neighborhood $N(u)$ in the graph. The update as mentioned in [Ham] can be represented as $h_u^{(k+1)} =$

$$\boldsymbol{h}_u^{(k+1)} = UPDATE^{(k)}(\boldsymbol{h}_u^{(k)}, AGGERGATE^{(k)}(\boldsymbol{h}_v^{(k)}, \forall v \in N(u))) \tag{2.1}$$

The functions UPDATE and AGGREGATE are differentiable and can be any type of neural networks. The superscripts are utilized to differentiate the embeddings and functions during the various iterations of message passing. To maintain high representational capacity and be trainable, an aggregator function should be symmetric, meaning that it is invariant to permutations of its inputs. Three possible aggregator functions are the Mean aggregator, LSTM aggregator, and Pooling aggregator [HYL17b].

2.2 Message passing in GCNN

Graph Convolutional Neural Networks (GCNNs) rely on message passing as the fundamental mechanism for information exchange and aggregation across the graph structure. Message passing in GCNNs is a recursive process in which each node iteratively receives and sends messages to its neighboring nodes before updating its own state based on the aggregated information.

At each iteration of message transmission, each node receives a message from each of its neighbors. The message is computed by combining the feature representation of the sending node with a learnable weighting matrix. The combination can be a simple dot product or a more complex operation such as a multilayer perceptron. The resulting message represents the information that the neighboring node wishes to transmit to the receiving node. After receiving messages from all neighbors, the node aggregates them by taking a weighted sum or maximum of the messages. The weights can be learned or predefined and represent the importance of each neighbor's message. The aggregated messages are then combined with the node's own feature representation using a different learnable weighting matrix to update the node's state. The updated state is passed on to the next iteration of message passing.

Overall, message passing in GCNNs allows each node to collect information from its neighbors and use it to update its own state. This mechanism can capture the local structure of the graph and disseminate information about it.

2.3 Zachary's karate club network

The Zachary's karate club network is a classic example of human interactions. The network has 34 nodes representing the members of a karate club and 78 edges showing each two individuals who had connections outside the club. A conflict between two members happens which leads to other members picking sides and creating two main groups in the network. The network is studied to predict which group each member joins after the conflict.

Figure 2.1 shows the members of the karate club network and their interactions.

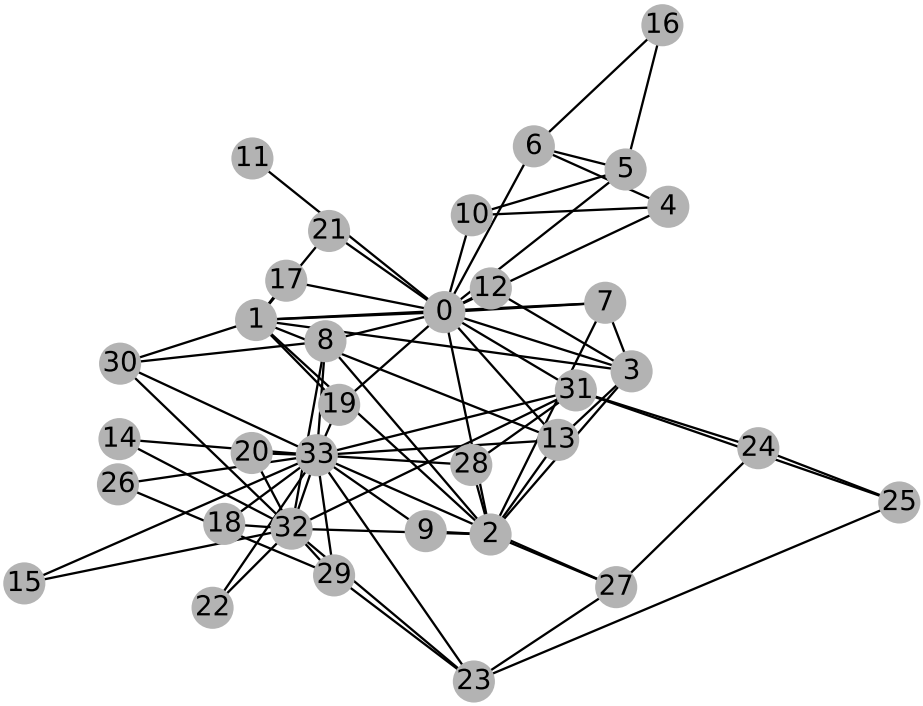


Figure 2.1: Zachary's karate club network

2.4 Modeling the karate club problem using GCNN

Given the karate club network and the two members who had a dispute; Assign each member to one of the two groups.

In graph theory this problem is known as a semi-supervised classification task where the label of two nodes are available and based on the network structure we try to predict the label for the rest of the nodes.

We use a GNN model to solve this classification task. The GNN model takes two inputs. The first input is the features of the nodes and the second input is the edge list from the karate club network. We use the identity matrix as the features for the nodes. The identity matrix is a square matrix with ones on the diagonal and zeros elsewhere. The GNN model will aggregate and transform the features from each nodes' neighbours and output the likelihood of each node being assigned to the two groups.

2.5 Training a GCNN

Since we are solving a classification task; we define the loss function as follows:

$$-\sum_{v \in V} y \log(f(X_v)) + (1-y)\log(1-f(X_v)) \tag{2.2}$$

Formula 2.2 calculates the negative likelihood of two nodes with their true values. Note that we calculate loss on the two nodes that their labels are given. We train the model with several epochs to achieve the final labels for all nodes. As you can see in Figure 2.2, the labels are randomly initialized in the first epoch. In Figure 2.3, the color assigned to each node is updated after 10 epochs. In the final epoch, we can see each node is classified and assigned the final label accordingly (Figure 2.4).

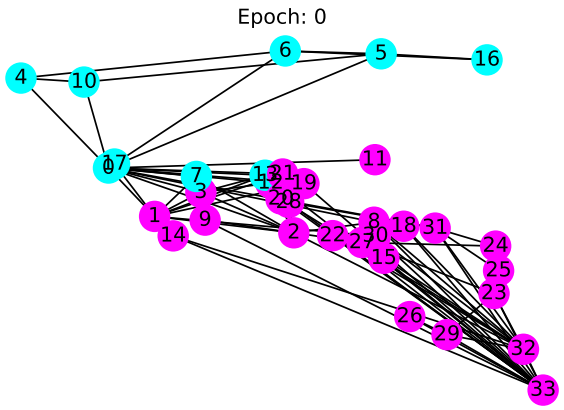


Figure 2.2: Training on the Zachary's karate club network, initialization

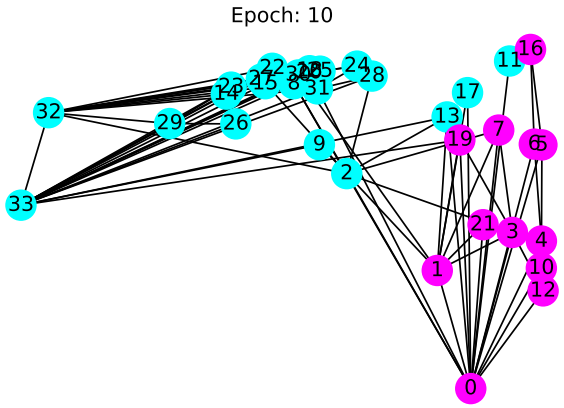


Figure 2.3: Training on the Zachary's karate club network, after 10 epochs

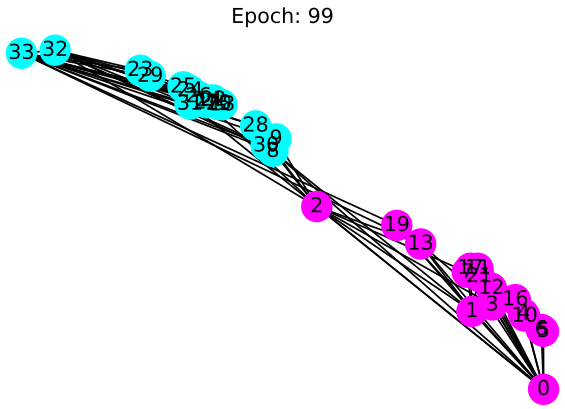


Figure 2.4: Training on the Zachary's karate club network, final epoch

3 Social Media Data Collection

In this chapter we introduce a system for collecting tweets through the Twitter API. We go through Twitter's API limitations and the tweet ID format. We show how it is possible to retrieve more tweets based on the tweet ID and describe the architecture of our proposed system. We conclude we the open challenges facing social media data collection.

3.1 Introduction

Twitter is a rich micro-blogging platform with live users generating content that can provide valuable data for research. The Twitter API gives access to tweets posted by users. Unfortunately, full data access is restricted to high-paying customers. We propose a system that can access otherwise unobtainable data. First, our system enables retrieving more live tweets than the free API returns. Second, it provides means to archive tweets that are older than a week which Twitter's API cannot do. We accomplish this by an informed prediction of tweet IDs generated based on the the Snowflakes service. Using our system with only 10 tokens, we are able to collect more than 35 million tweets/week, that were not part of the free tweet stream provided by Twitter. Our system enables research and deeper analysis of Twitter trends or historic events by providing large amounts of Twitter data.

There are many online social platforms for people to express themselves, share their thoughts and interact with one another. Thus, online social networks have become a massive source of information spreading and diffusion. In the past decade we have seen a huge growth of user generated content on online social networks. Twitter as a micro-blogging service has become a medium for users to share information and news. It allows communication through 280 character long messages and comments. These short messages (similar to an SMS) are called tweets and made the company, founded in 2006, world famous. After Twitter's IPO in 2013, it has grown very fast and now has 200 million daily users worldwide [22b]. Various research groups have

collected this data for topic modeling, community detection, user personality and behavior analysis etc. However collecting data from Twitter's API is limited to a 1% sample of all the tweets. We show it possible to collect a higher sample of tweets based on the Snowflake algorithm and the format of tweet ids.

To enable in-depth investigations of tweets, Twitter offers a live stream of its data as a commercial product called Twitter Firehose. This service is said to cost a 6-digit sum monthly according to forum entries [3,4]. Nevertheless, it is used by some management consultancies, security companies and political offices. Twitter also periodically provides large data sets that can be used to conduct academic research. These datasets usually comprise a week's worth of Twitter's total traffic. However, there is a great deal of interest for academic research in a continuous Twitter livestream, as it can provide daily updated data that exceeds the available data packages from Twitter after only a few weeks. Similarly, there is great interest in near-complete data sets on a specific time period (e.g., during elections or natural disasters).

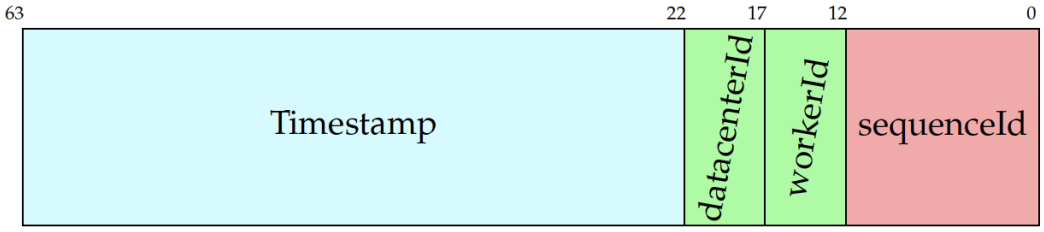


Figure 3.1: Tweet ID Binary Format

The Twitter Application Programming Interface (API) has seen several changes since the micro-blogging website was first launched in 2006. Before 2009, it was possible for research groups to crawl tweets on Twitter at a large scale. For example [Kwa+10] collected 106 million tweets and 41.7 million user profiles. However in September 2009, Twitter announced a new terms of service with a new rate-limit for the API. Depending on the API function an authorized user can make a specific number of requests in a 15 minutes window. For example for collecting followers' ids 15 requests and for getting users' profile 900 requests can be made before you hit the rate limit.

In this chapter, we investigate the following questions:

1. How can we collect a bigger sample of tweets based on the tweet id format?

2. With how many API keys can we collect a higher sample of tweets from Twitter's timeline?

3. How can we build a system that enables retrieving more live tweets?

3.2 Motivation & Background

Twitter offers a variety of APIs to interact with their systems. Among others, there are live streams that output part of all or even all tweets. These APIs can be used to discover new trends or do analysis on the live data. Twitter's main streaming endpoints are the following:

- **1% sampled stream** The 1% sampled stream allows all interested users to get a picture of what is currently happening on Twitter. It provides what appears at first glance to be a random 1% fraction of all tweets. Jürgen Pfeffer et. al. found out that Twitter filters tweets based on their creation timestamp [PMM18]. All tweets processed by the Twitter servers having a millisecond in the range from 657 to 666 are routed to the 1% sampled stream (see Fig. 3.2).

- **10% academic stream** This stream is only available to selected researchers and cannot be found on Twitter's websites. The stream outputs all tweets from milliseconds 657 to 756 [PMM18].

- **10% enterprise stream** Like the academic stream, the enterprise stream, also called *Decahose*, outputs 10% of all tweets. Even though the number of tweets is very similar, different millisecond ranges are streamed. For the Enterprise stream, all tweets that have a millisecond with the format X0X are transmitted [PMM18].

- **Filtered Stream** This endpoint, which can only be accessed with a developer token, enables filtering of the live feed. With a simple developer token 5 filter rules can be formulated. If you apply for 'larger' developer tokens or the academic research token, the number of filter

rules increases up to 1000. If the filter criterion is not sufficiently restrictive, a maximum of 1% of all tweets will be returned. The returned tweets from filtered stream also count towards the monthly tweet cap.

- **Firehose Twitter** Firehose was the API endpoint that allowed enterprises to stream tweets for a huge amount of money. All tweets processed by Twitter will be routed to the Firehose stream. This service was discontinued with the acquisition of Gnip in 2014.
- **Lookup Endpoint** The Lookup endpoint allows the request of a tweet by its Snowflake (detailed structure is described in section 3) and returns the complete tweet object in case the tweet exists. Packages of 100 tweet-ids can be requested per GET request.

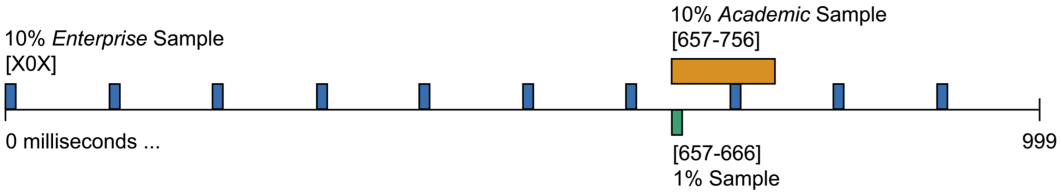


Figure 3.2: Sampling based on millisecond windows (Ill. by Jürgen Pfeffer et. al. [PMM18])

When trying to conduct research of different types with the help of Twitter, most scientists are left with Twitter's 1% sampled stream. Even though it delivers about 60 tweets per second as of February 2022, this is not enough to get a complete overview of all current events.

For various reasons, there may be an over- or under-representation of certain accounts [PMM18]. This can happen randomly or with an attempt to influence exactly those people who use the 1% sampled stream. Looking for ways to get more tweets, we took a closer look at the tweet-ids, the so-called Snowflakes. In the process, we realized that these ids are not generated as randomly as they seem at first glance. In combination with the Lookup API and the considerations of Jason Baumgartner, we built a software system that outputs tweet ids with a high hit rate, which we can then query and store.

The Snowflake algorithm generates unique ID numbers at high scale. It guarantees that the ids are k-sort-able and can be computed in a distributed

Listing 3.1: Snowflake code, IdWorker.scala

```
1           ...
2           val twepoch = 1288834974657L
3
4           val workerIdBits = 5L
5           val datacenterIdBits = 5L
6           ...
7           val sequenceBits = 12L
8
9           val workerIdShift = sequenceBits
10          val datacenterIdShift = sequenceBits + workerIdBits
11          val timestampLeftShift = sequenceBits + workerIdBits
12          + datacenterIdBits
13          ...
14          def nextId(): Long = synchronized {
15                  ...
16                  ((timestamp - twepoch) << timestampLeftShift) |
17                  (datacenterId << datacenterIdShift) |
18                  (workerId << workerIdShift) |
19                  sequence
20          }
```

fashion. Twitter uses these as ids for tweets. It is a 64-bit composite id made up of 42 bits of timestamp in ms, 5 bits of datacenter id, 5bits of worker id and 12 bits of sequence id (Fig. 1). Since the timestamp is located in the most significant bits, Snowflakes can be k-sorted by time.

Important evaluations of the structure of Snowflake's and the distribution of its individual components were described by [PMM18]. The description of the different streaming endpoints helped in our analyses and showed that the sampled stream does not necessarily provide a random representation of all tweets.

The code 3.1 is implementing the Snowflake algorithm, a unique ID generator used by Twitter to generate tweet IDs. It defines several constants to configure the number of bits allocated for different parts of the ID generation process, including the worker ID, datacenter ID, and sequence number. The nextId() function generates a unique ID by combining a timestamp with

the worker ID, datacenter ID, and sequence number using bit shifting and bitwise OR operations. The twepoch variable is a reference point in time (in milliseconds since the Unix epoch) that is used as the starting timestamp for ID generation. The synchronized keyword ensures that the nextId() function can only be executed by one thread at a time to avoid ID collisions.

3.3 Methodology

Baumgartner showed that in practise, Snowflake's service consist of very few different datacenter and server ids. Moreover, these ids rarely change over time. The sequence id follows a pattern as well. To confirm the assumptions, we streamed the 1% Sampled Stream for about 6 days between Tue Jan 18 2022 21:14:07 GMT+0000 to Mon Jan 24 2022 19:58:40 GMT+0000 and saved the tweet-ids. This is the data that the following analysis is based on. As mentioned in section 2, Twitter provides the Lookup API that can be used to retrieve tweets by tweet id. Any Twitter user can authorize a Twitter app that allows the app to make 900 requests per 15-minute interval to this API on behalf of the user, which is exactly one request per second. Given the structure of Snowflakes it is easy to see that for each datacenter, server, sequence id combination, there are only 1000 possible snowflakes, one for each millisecond. In order to lookup all milliseconds for a given triplet, we need to make 10 API calls with 100 Snowflakes each. If this is supposed to happen live and continuously, we need 10 user tokens per triplet. Given our 6-day data, we can see that not all triplets are equally likely to exist.

Figure 3.3 shows the distribution of tweet ids over these triplets. Thus, we can grab about 1.24% of all tweets with 10 user tokens, depending on the current workload of Twitters servers.

Using this method, the Snowflakes can be guessed. In some cases, a tweet with this guessed snowflake exists and is then returned via the Twitter-API. Baumgartner uses several example calculations to show the number of tokens that can be used to achieve various coverage of the live stream. Since these calculations are outdated, we recalculated the numbers based on our data. Figure 3.4 and Table 3.1 show how many API tokens we need for a given live percentage of tweets.

	Sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Datacenter	Server																			
13	0	1,235%	0,897%	0,649%	0,447%	0,288%	0,209%	0,145%	0,097%	0,063%	0,040%	0,024%	0,015%	0,009%	0,005%	0,003%	0,002%	0,001%	0,001%	0,000%
13	2	1,236%	0,897%	0,649%	0,449%	0,288%	0,208%	0,144%	0,096%	0,063%	0,039%	0,025%	0,015%	0,009%	0,006%	0,003%	0,002%	0,001%	0,001%	0,000%
13	3	1,240%	0,899%	0,647%	0,447%	0,289%	0,209%	0,143%	0,097%	0,063%	0,039%	0,024%	0,015%	0,009%	0,005%	0,003%	0,002%	0,001%	0,001%	0,000%
13	4	1,241%	0,901%	0,650%	0,445%	0,288%	0,207%	0,143%	0,096%	0,063%	0,039%	0,024%	0,015%	0,009%	0,005%	0,003%	0,002%	0,001%	0,001%	0,000%
13	5	1,240%	0,900%	0,649%	0,449%	0,289%	0,208%	0,143%	0,097%	0,063%	0,039%	0,024%	0,015%	0,009%	0,005%	0,003%	0,002%	0,001%	0,001%	0,000%
13	7	1,237%	0,899%	0,653%	0,448%	0,287%	0,208%	0,143%	0,097%	0,063%	0,038%	0,025%	0,015%	0,009%	0,006%	0,003%	0,002%	0,001%	0,001%	0,000%
13	8	1,242%	0,899%	0,651%	0,447%	0,289%	0,208%	0,144%	0,096%	0,062%	0,039%	0,025%	0,015%	0,009%	0,005%	0,003%	0,002%	0,001%	0,001%	0,000%
13	9	1,243%	0,899%	0,651%	0,446%	0,288%	0,208%	0,143%	0,097%	0,061%	0,039%	0,025%	0,015%	0,009%	0,006%	0,003%	0,002%	0,001%	0,001%	0,000%
11	4	0,842%	0,747%	0,654%	0,551%	0,441%	0,356%	0,283%	0,216%	0,164%	0,120%	0,085%	0,060%	0,042%	0,028%	0,019%	0,012%	0,008%	0,006%	0,004%
11	5	0,823%	0,734%	0,647%	0,548%	0,439%	0,361%	0,288%	0,224%	0,169%	0,125%	0,089%	0,063%	0,044%	0,031%	0,021%	0,014%	0,009%	0,006%	0,004%
11	6	0,844%	0,748%	0,655%	0,549%	0,440%	0,356%	0,281%	0,217%	0,162%	0,119%	0,085%	0,060%	0,042%	0,028%	0,019%	0,013%	0,008%	0,006%	0,004%
11	17	0,838%	0,740%	0,657%	0,551%	0,439%	0,358%	0,283%	0,217%	0,166%	0,120%	0,087%	0,061%	0,042%	0,029%	0,019%	0,013%	0,009%	0,006%	0,004%
11	18	0,839%	0,742%	0,654%	0,550%	0,442%	0,356%	0,281%	0,216%	0,163%	0,121%	0,086%	0,061%	0,043%	0,029%	0,019%	0,013%	0,009%	0,006%	0,004%
11	19	0,837%	0,746%	0,654%	0,551%	0,441%	0,358%	0,282%	0,217%	0,163%	0,120%	0,087%	0,060%	0,042%	0,029%	0,019%	0,013%	0,009%	0,006%	0,004%
11	26	0,843%	0,748%	0,652%	0,551%	0,441%	0,358%	0,284%	0,215%	0,162%	0,118%	0,084%	0,060%	0,041%	0,028%	0,019%	0,013%	0,008%	0,005%	0,003%
11	27	0,821%	0,736%	0,648%	0,545%	0,442%	0,359%	0,286%	0,221%	0,169%	0,123%	0,090%	0,063%	0,044%	0,031%	0,021%	0,014%	0,009%	0,006%	0,004%
11	28	0,839%	0,745%	0,657%	0,550%	0,440%	0,359%	0,279%	0,216%	0,163%	0,120%	0,086%	0,060%	0,041%	0,028%	0,019%	0,012%	0,008%	0,005%	0,003%
11	29	0,843%	0,747%	0,656%	0,553%	0,438%	0,357%	0,283%	0,215%	0,162%	0,119%	0,085%	0,060%	0,041%	0,028%	0,019%	0,012%	0,008%	0,005%	0,003%
10	5	0,609%	0,477%	0,345%	0,226%	0,137%	0,088%	0,057%	0,038%	0,024%	0,015%	0,010%	0,007%	0,004%	0,003%	0,002%	0,001%	0,001%	0,000%	0,000%
10	6	0,529%	0,425%	0,332%	0,239%	0,164%	0,113%	0,078%	0,053%	0,037%	0,024%	0,016%	0,011%	0,008%	0,005%	0,004%	0,003%	0,002%	0,001%	0,001%
10	7	0,607%	0,479%	0,345%	0,227%	0,138%	0,087%	0,057%	0,037%	0,024%	0,016%	0,010%	0,006%	0,004%	0,003%	0,002%	0,001%	0,001%	0,000%	0,000%
10	17	0,610%	0,479%	0,348%	0,225%	0,138%	0,089%	0,057%	0,038%	0,025%	0,016%	0,010%	0,006%	0,004%	0,003%	0,002%	0,001%	0,001%	0,000%	0,000%
10	18	0,527%	0,428%	0,333%	0,240%	0,163%	0,113%	0,077%	0,053%	0,036%	0,024%	0,017%	0,011%	0,008%	0,005%	0,004%	0,002%	0,001%	0,001%	0,001%
10	20	0,611%	0,480%	0,347%	0,227%	0,137%	0,086%	0,057%	0,037%	0,024%	0,015%	0,010%	0,007%	0,004%	0,003%	0,002%	0,001%	0,001%	0,000%	0,000%
10	21	0,608%	0,479%	0,347%	0,225%	0,137%	0,087%	0,056%	0,036%	0,024%	0,015%	0,010%	0,007%	0,004%	0,003%	0,002%	0,001%	0,001%	0,000%	0,000%
10	23	0,605%	0,480%	0,346%	0,228%	0,138%	0,088%	0,057%	0,037%	0,024%	0,016%	0,010%	0,006%	0,004%	0,003%	0,002%	0,001%	0,001%	0,001%	0,000%
10	24	0,606%	0,482%	0,346%	0,227%	0,138%	0,088%	0,056%	0,036%	0,024%	0,016%	0,010%	0,007%	0,004%	0,003%	0,002%	0,001%	0,001%	0,000%	0,000%
10	25	0,606%	0,477%	0,347%	0,229%	0,139%	0,088%	0,056%	0,037%	0,024%	0,016%	0,010%	0,007%	0,004%	0,003%	0,002%	0,001%	0,001%	0,000%	0,000%

Figure 3.3: Percentage of tweets processed by each datacenter, server, sequence-id triple

3.4 Snowflake Analysis

Given the Snowflake data from our 6-day period, we can run some analyses. First of all, we tried to verify the calculations of Baumgartner. In doing so, we quickly encountered large differences in the results. This could be explained by structural changes in Twitter's data centers, which was necessary due to the immense growth or seasonal differences, or a bad sampling in Baumgartner's data.

- **Timestamps** As described in section 3.2, the sampled stream returns all tweets that are published in the millisecond range between 657 and 666 which equals to 1% of all tweets on average. Our data could verify that.
- **Datacenter and Server ID** According to our data, Twitter currently operates 28 servers, located in a total of 3 datacenters for their Snowflake generation. During the time of our stream analysis, there was no change in the number of servers and their ids. Even after several weeks, the ids remained the same.
- **Sequence IDs** The most interesting information for us is the distribution of the sequence ids. These are the only remaining unknown after

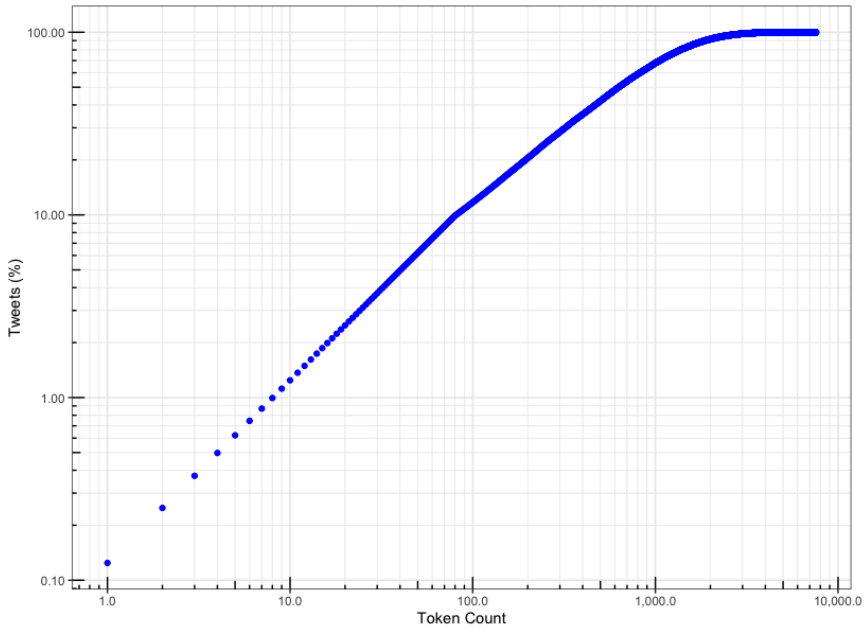


Figure 3.4: Percentage of tweet-coverage with a given number of User-Token

determining the list of active servers. The distribution of sequence ids resembles an exponential decay. This means that the first id makes the most frequent appearance and each subsequent id is used by fewer and fewer tweets. The following table shows the first 10 Sequence ids with the number of tweets in our dataset that use them and the percentage of all tweets from our sample. Figure 3.5 shows the number of tweets that are assigned to a specific sequence id.

After we identified both the list of servers and the distribution of sequence ids, we linked the two pieces of information. This resulted in an overview that shows the number of processed tweets for each datacenter - server - sequence id combination. In Figure 3.3, you can easily see which servers belong to which data center. The servers within a datacenter have a very similar distribution of sequence ids and the percentage of processed tweets. However, the data centers themselves have very different workloads and even with the same sequence id there are large differences.

Percentage	Needed Tokens
1%	8
5%	41
10%	81
20%	195
50%	629
90%	1872
95%	2342
99%	3402
99.9%	5022
99.97%	6788

Table 3.1: Percentage of tweet-coverage with a given number of User-Token

- **Server Activity** Figure 3.6 shows an overview of all servers in all data centers during our data collection. The segmentation into the three data centers can again be seen very clearly. After a little more than two days of data collection, there were two major downtimes of individual datacenters. The otherwise busiest datacenter processed very few tweets for about 6 hours. During the downtime, another datacenter immediately stepped in and took over all tweets from the outaged one. There was no distribution between the two remaining data centers. We observed the same behavior during the downtime of another data center. In the bottom most line chart in Figure 3.6, we can still see that the total number of tweets processed did not change during the downtimes. A roughly sinusoidal wave motion in datacenter activity can also be seen in the line chart. This appears to be the day-to-day usage of Twitter in different regions of the world. The datacenters with id 10 and 13 appear to be geographically close to each other due to the very similar utilization lines.

3.4.1 Further Improvements

In previous section, we have shown how the analysis of the snowflakes has an impact on the hit rate when creating the tweet-ids. By choosing the best

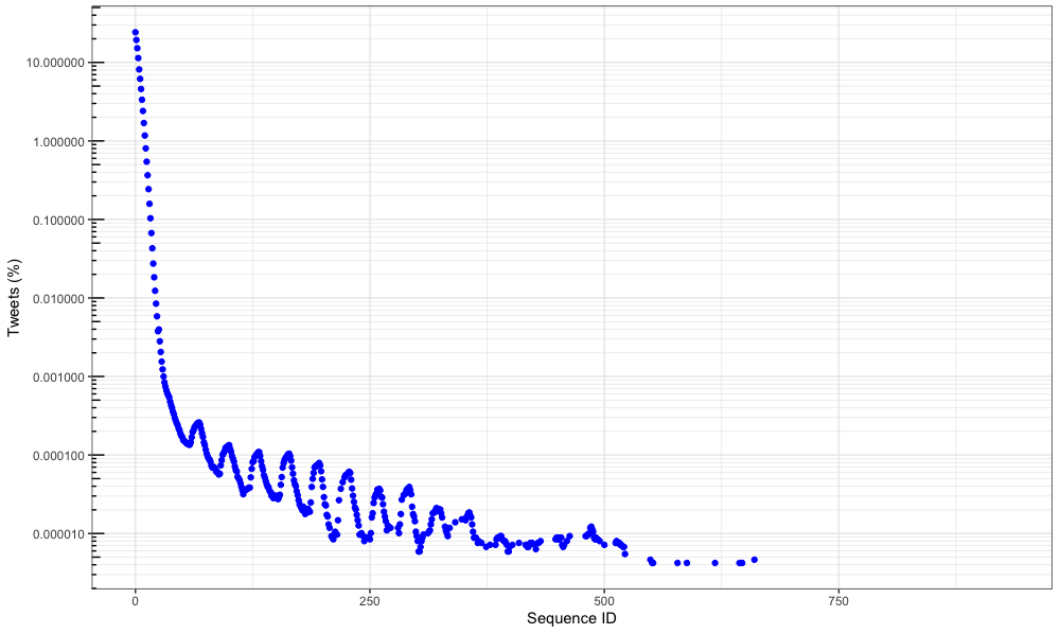


Figure 3.5: Percentage of tweets per Sequence-ID

performing server, we can sometimes collect twice as many tweets as with the worst performing server with the same sequence id. However, in Figure 3.7 we see that it is not always one specific datacenter that consistently processes the most tweets. For example, the most active datacenter has the id 13 and experiences a downtime of about 3 hours on January 21, 2022, between 9 am and 12 pm CET. With a static determination of the best server, we would therefore not be able to collect a single tweet during this time. The same applies due to the different workloads resulting from the geographical position and the local time of the datacenters. The data center with the id 11 is more active between 6 p.m. and midnight CET than the overall most active data center with the id 13. With this knowledge we have built our system in such a way that the guessing of the snowflakes is not done with a static table, but dynamically. For each 10 minute time window we store the number of tweets sent by each datacenter, server, sequence id combination over the 1% Sampled Stream. We then use the best combinations for our requests depending on the number of tokens available to us. In this way, we also avoid querying sequence id 0 at the datacenter with the fewest tweets when

Sequence ID	Tweets	Percentage	Cumulative Percentage
0	5741186	24.199%	24.119 %
1	4581115	19.309%	43.428 %
2	3599038	15.170%	58.598 %
3	2697723	11.371%	69.969 %
4	1930844	8.138%	78.107 %
5	1463997	6.170%	84.277 %
6	1087688	4.584%	88.861 %
7	794253	3.347%	92.208 %
8	571228	2.407%	94.615 %
9	400903	1.689%	96.304 %

Table 3.2: Sample Stream Coverage with first 10 Sequence ids

sequence id 1 at the datacenter with the most tweets has a higher chance of being a hit. Datacenter downtimes are also avoided this way and we waste as few queries as possible.

3.5 System Architecture

The goal of our system is to download as many tweets as possible in an unbiased way given a certain number of tokens. The number of tokens limits the number of API calls we can make, so the system must make sure that we perform requests at a rate compatible with our tokens. There are multiple ways one could go about this.

It makes sense to know how many potential Snowflakes we can check in a given time, so that we do not generate more than actually needed. That is why we chose to have one component that inherently rate-limits the Snowflake-guess generation by design called the Task-manager. It looks at the distribution of server ids, datacenter ids, sequence ids and generates Snowflakes based on the most active combinations in order to maximize the hit rate of supposed Snowflakes. These potential Snowflakes are then pushed to a queue where workers will take them, use the OAuth tokens to fetch the content and metadata for them, and save the data into a database. In order to acquire this distribution of ids, another component, the Analysis worker,

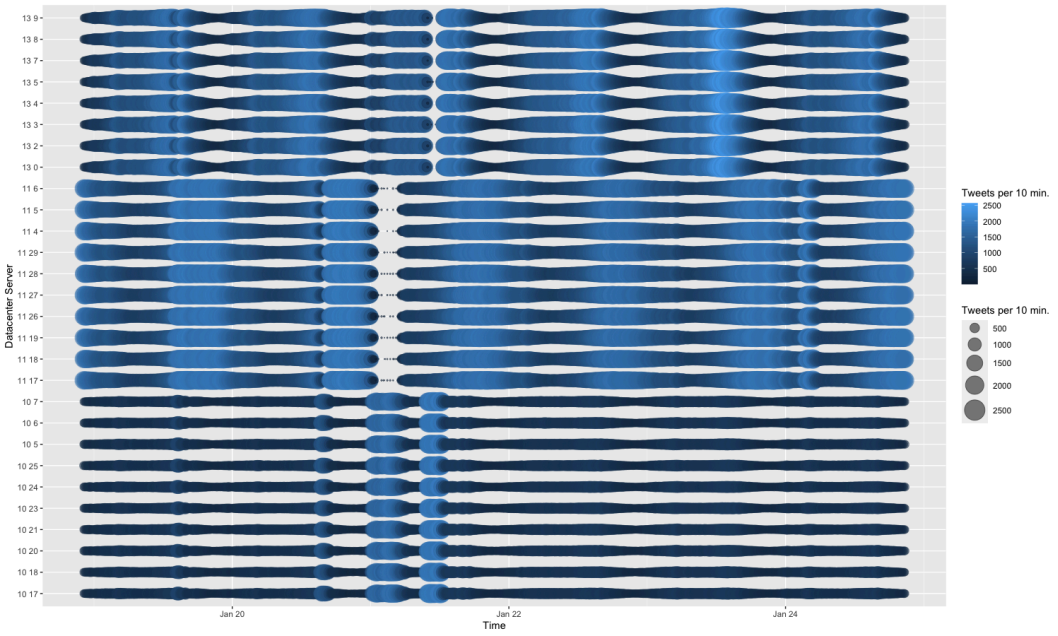


Figure 3.6: All servers in all three data centers and their utilization over time

is continuously observing the 1% sampled stream that Twitter provides. It decodes the Snowflakes, counts the occurrences of the id-triplets and saves them for every ten-minute window. Since the 1% are selected by the ms range of the Snowflakes, there should not be any bias towards any datacenter, worker or sequence id. This data is then used by the Task-manager for choosing the best combinations.

3.6 Implementation

We chose to develop the backend based on dockerized microservices. Services communicate via HTTP requests if necessary and move data through a RabbitMQ message queue. Our system consists of the Core Components as well as Of-The-Shelf Components. After a brief look at our ReactJS9 webapp, we will take a detailed look at the backend services that are responsible for retrieving the tokens.

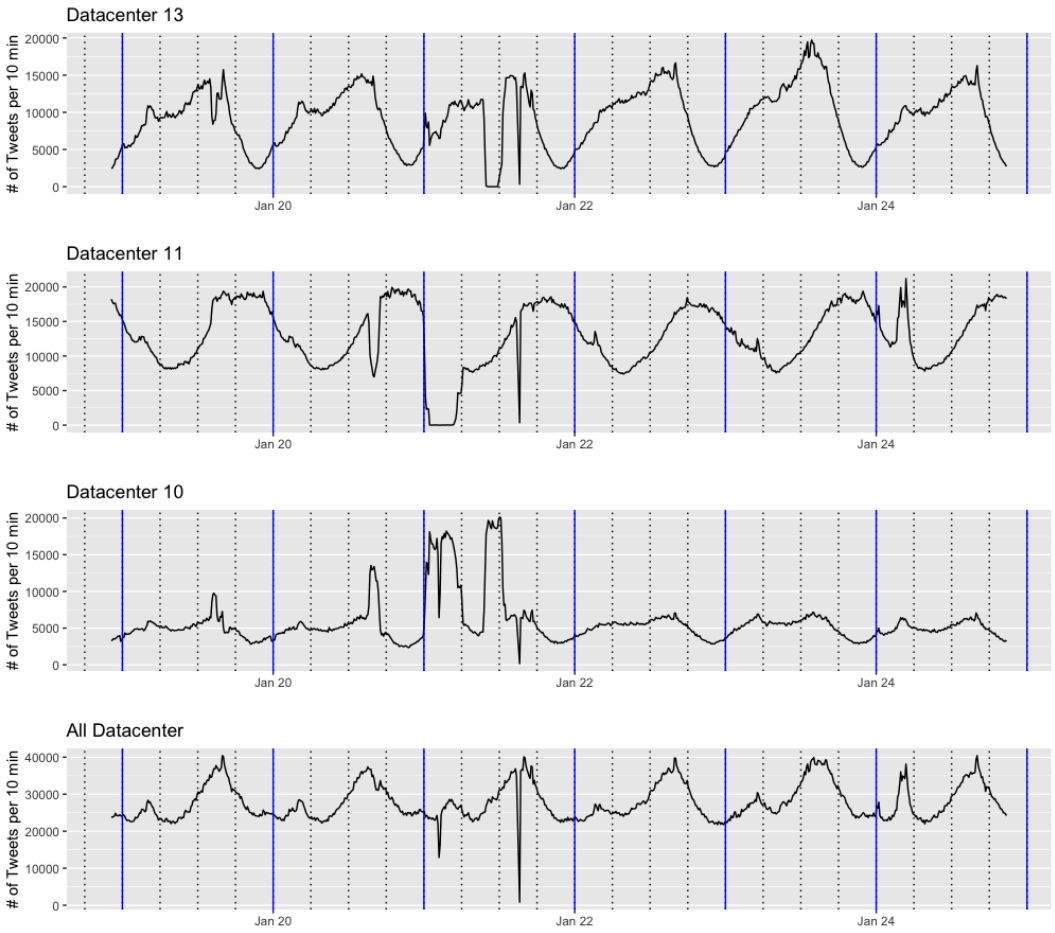


Figure 3.7: Data center utilization over time

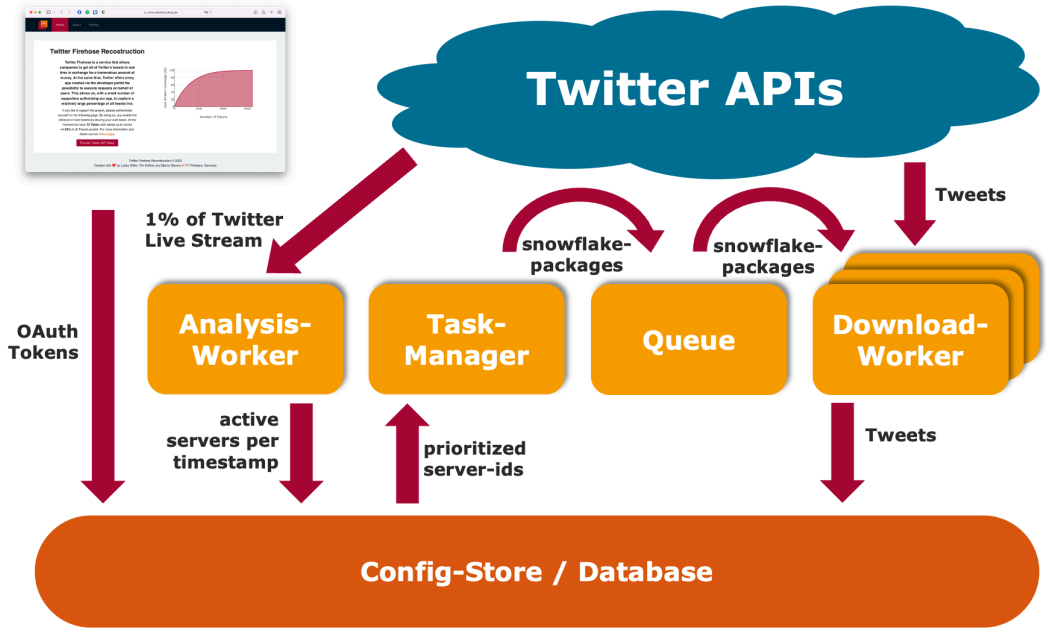


Figure 3.8: System Architecture, red arrows indicate data flow

3.6.1 Webapp

The webapp will serve as the project's public website, encouraging users to donate their OAuth tokens. It also manages the OAuth workflow, where the user logs into Twitter and authenticates the app to use their token. In addition, the webapp handles a token storage and makes it available in the form of an API.

- **Structure** The webapp includes several pages, the most important of them is the landing page. This page directly allows authenticating our app on Twitter for donating the Oauth token. Moreover, a diagram shows how many tweets can be covered with the current and future number of tokens. In addition to the present homepage, there is the about page, which goes into more detail about how our system works, giving the user the assurance that we are not abusing their trust. Additional pages for the privacy policy, for authentication errors, for reporting a successful authentication as well as a dashboard have already been created, but only implemented in parts.

- **Token Collection** The authentication of users at Twitter is done by an OAuth 1.0 flow. In this process, the visitor of our website is redirected to Twitter and has to confirm the authorization there after successful login. Twitter then sends the user back to our website together with the OAuth tokens. These are stored in a JSON file and can be accessed via the credentials/credentials.json route. Access is thereby secured with basic auth and SSL.

3.6.2 System Components

The SampledStreamFollower is responsible for extracting the distribution of ids in live Snowflakes from the sampled stream. The Taskmanager is responsible for generating Snowflakes and inserting them into a queue, from where an instance of a Downloadworker takes them. The worker then attempts to fetch metadata for these ids and saves those tweets that actually exist into a database. In order to provide a central instance governing the API tokens and their rate limits, the workers request the token from the Credentialsmanager. It ensures only valid, non-revoked, non-suspended tokens reach the workers. Below, we will describe each of our components in more detail.

- **SampledStreamFollower** The SampledStreamFollower continuously listens to the free sampled stream endpoint where Twitter provides about 1% of all tweets. It counts occurrences of the triplets (server id, datacenter id, sequence id) and saves them into a key-value store, Redis in this case. Since the sampled stream endpoint returns very little information with the tweets, this component buffers the Snowflakes and pushes packets of 100 ids into a message queue, RabbitMQ in our case. This will allow us to retrieve the tweets with all the fields later.
- **Taskmanager** The Taskmanager generates Snowflakes on a best-effort basis. It takes into account the distribution of server ids in the current 10-minute window in order to maximize the chance for a hit. This distribution can be seen in Figure 4. As we need to make sure to not guess more ids than we can probe given our current number of available tokens, the Taskmanager gets the number of tokens from the Credentialsmanager and sets the rate accordingly. Since the Taskmanager is

responsible for not overwhelming the workers, every message gets a time-to-live (TTL) of one hour. In case of workers failing or falling behind, this ensures that workers can catch up again. Without the TTL, the worker side could experience a sustained and much higher rate than intended from the Taskmanager. This would result in workers not catching up, thus not being live anymore.

- **Downloadworker** The Downloadworker fetches a list of 100 Snowflakes from a rabbit message queue and uses the LOOKUP endpoint of Twitters APIv1.1 to download the tweet metadata for them in a JSON format. It is a scalable component, the number of instances can be adjusted to the number of available tokens. Using one worker per token is a good rule of thumb for the start, although on a capable system, one worker is able to perform more than one fetch per second. Since workers sometimes need to retry a failed operation, it makes sense to have some spare workers available.

- **Credentialsmanager** When workers need API credentials for their lookup calls, they request them from the Credentialsmanager. This manager fetches an up-to-date list of credentials from the frontend and verifies that they (still) work. Upon a workers request, it returns a valid token and caches the workers id. If this same worker requests a new token within the 15 minute rate limit window, the former token is considered to be empty until 15 minutes have passed. Thus, that token will not be returned to any other worker either.

In order to ensure a complete usage of the available tokens, the Credentialsmanager currently serves multiple workers the same token if it is still valid. Since following the ids from the sampled stream ($\tilde{6}0$/s) only needs less than one token, the rest would be wasted if no other worker were to use the same token during the same window. A similar situation can occur if a machine is slow: if a worker can process less than one request per second, it would also leave parts of a token unused. In scenarios where this is not the case or where tokens clearly outnumber the workers, a different strategy might be better suited.

3.7 Conclusion

In this chapter we studied Twitter's Snowflake algorithm for generating unique tweet IDs. The Twitter API is limited to a small sample of tweets. We show that based on the binary representation of tweet ids and with access to multiple API keys it is possible to circumvent this limitation. We showed that massive tweet collection by informed snowflake prediction is possible and practical. Our documented system is easily deployable, scalable and runs reliably. You can access the code from our GitHub repository (https://github.com/saaay71/twitter-firehose-reconstruction). We used the sampled stream for estimating the distributions instead of sampling the space as a faster, more consistent and more reliable alternative. Moreover, we updated the estimation of necessary tokens for given percentages of the twitter stream for 2022. Actually recreating a 99% Firehose stream will take a lot of user tokens and will most likely be noticed by Twitter, possibly resulting in counter-measures to this method. Revoking the app token that was used to acquire the users OAuth tokens will already make the endeavor much harder. However, our framework can still be used successfully by researchers from all fields if a specific event within a certain time should be analyzed. Since access to a substantial amount of tweets is only possible via Twitter-selected data dumps or following live APIs, it is very hard to analyze events that happened more than a week ago. With our method and a fair number of tokens, a 24h window of tweets can be fetched over a longer course of time, whereby there are then significantly more tweets available than from the sampled stream. We know that the sampled stream is not representative of the Firehose due to bad actors that target this time slice [PMM18]. Because we use the sampled stream for calculating the most active servers, this leads to a slight overestimating of the expected hit rate of snowflakes, since there will be less tweets for sequence id 0 in the rest of the Firehose. However, this should have no impact on the general distribution of servers.

4 User Engagement Prediction On Online Social Media

This chapter presents our approach to predicting user engagements for the ACM RecSys 2020 challenge. The goal of the challenge was to predict whether a user would engage with an tweet in one of four ways: like, retweet, reply or qoute. As part of this challenge Twitter released an official dataset. This dataset is used for the trained model in this and the next chapter. We analyzed the provided dataset and derived several observations, which were then implemented in our final model. In this chapter, we approached the challenge as a binary classification problem, where we trained a gradient boosting classifier for each engagement type.

4.1 Introduction

Most social media websites make use of recommender systems to show the content of interest for their users and to keep them engaged with the platform. On Twitter users can share and engage with the content by tweets. The ACM RecSys challenge 2020 focuses on predicting tweet engagements by users. In this chapter, we present our approach for modeling this task. We take a deep look into the dataset and provide interesting observations from the dataset. Based on these findings, we construct a set of features and use gradient boosting trees to classify different types of engagements.

Twitter as a social media and microblogging service has made it possible for online users to share and engage with their content in the form of tweets. Users can decide between two different approaches on how these tweets are presented to them on their timeline. The first way is based on time where the most recent tweets will be shown and the second method can algorithmically rank the tweets based on a measure of interest and potential engagement [Bel+20]. As users are looking for new and interesting content, recommender systems can play a big role to help users find something interesting and engage with tweets.

The recommender system (RecSys) challenge 2020 [20] is organized for the

task of predicting user engagements on tweets. As part of this competition, a dataset of 200 million engagements is released which consists of two weeks of tweets posted on the Twitter timeline and the engagements made by the users on the tweets. The final goal is to predict the probability of four possible engagement types with a tweet. The engagements with tweets include liking, replying, retweeting and quoting (retweeting with comment).

In this chapter, we investigate the different features of the dataset and show several patterns observed within the data itself. In the next step, we take a classical machine learning approach by feature engineering and training models to classify the different types of engagements. We define the set of features based on the raw features available in the data and our observations. For each engagement, we model the task as a binary classification problem using gradient boosted trees. There are two main metrics taken into account to evaluate the models: the precision-recall curve (PR-AUC) and the relative cross entropy (RCE). The final ranking score on the leaderboard is calculated based on the average values of RCE and PR-AUC across the four engagement types. The generated figures and source code of our method is available online[1].

The rest of this chapter is organized as follows. First, we provide the details about the released dataset and present the different patterns and observations in the dataset. In Section 3, we state our overall methodology for extracting features and training the models. Section 4, describes the evaluation metrics and the results of our approach. As the dataset provided has data that is not available from the Twitter API, we discuss the possible future work on this dataset in Section 5. Finally, we conclude the paper in Section 6.

4.2 The Dataset

As part of the ACM RecSys challenge, Twitter released a dataset of roughly 200 million engagements of users on tweets. The data, spans for two weeks between 6 February 2020 until 19 February 2020. The first week was released as the training dataset and the second week was divided equally for the validation and testing dataset.

This dataset is unique as it contains information that can not be retrieved

1 https://github.com/saaay71/RecSys-Challenge-2020

from the Twitter API [2]. For example, it is possible to collect the public engagements of public user profiles. However the information regarding what items a user has observed and decided not to show any engagement can not be retrieved from the API. In this dataset, such data is provided with a special attention given to privacy. As revealing such data would be a privacy leak, the dataset is mixed with data that a user might not have seen at all. Therefore we have pseudo-negative number of tweets that a user was not interested in [Bel+20].

Each engagement record of the dataset consists of four main parts: Tweet features, Engager features (features regarding the user who created the tweet), Engagee features (features regarding the user who engaged with the tweet), Engagement features (if the user engaged with the tweet, the timestamp of engagement is provided). For detailed description of the features and their types see [Bel+20]. Table 5.1 shows a summary of training, validation and test datasets.

		Training	Validation	Testing
Dataset size		69G	6.9GB	5.8GB
Number of records		121M	12.4M	12.4M
Number of unique tweets		57.7M	8.67M	8.67M
Number of unique users		28.2M	9M	9M
Timestamp of first tweet		2020-02-06 00:00:00	2020-02-13 00:00:00	2020-02-13 00:00:00
Timestamp of last tweet		2020-02-12 23:59:59	2020-02-19 23:59:59	2020-02-19 23:59:59
Tweet type distribution	Top_level	57.9%	57.8%	57.8%
	Retweet	33.2%	33%	33%
	Quote	8.9%	9.2%	9.2%

Table 4.1: The general statistics about the dataset (rounded down)

2 https://developer.twitter.com/en/docs/api-reference-index

Patterns and Observations in the Dataset

4.2.1 Flow of engagements

Figure 4.1 shows the different types of engagements on a hourly basis. We can see that tweets are engaged in the following order: like (74.9%), retweet (19.2%), reply (4.6%) and quote (1.3%). The ratio of engagements is roughly constant over time. As we will see in the evaluation section, this observation can help compare any proposed model with this baseline model that solely predicts based on the flow of engagements.

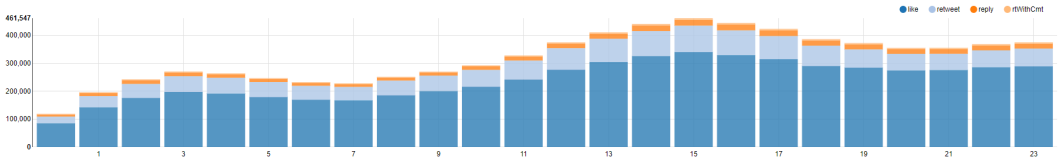


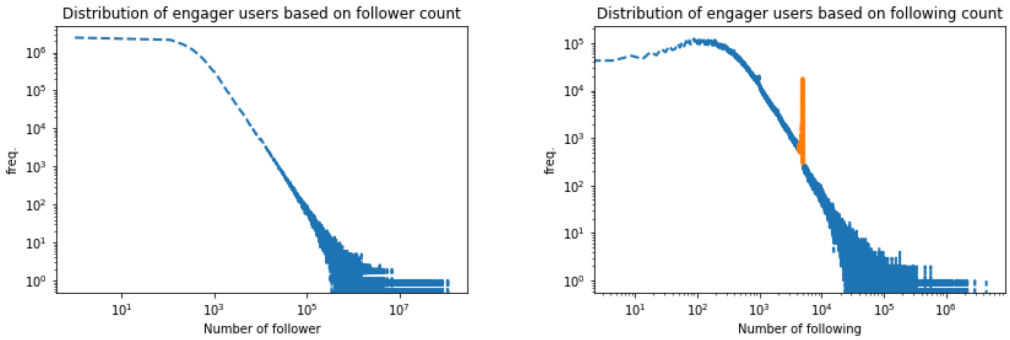
Figure 4.1: The amount of different types of engagements on all tweets in the first 24 hours.

4.2.2 Follower and following distribution

We take a look at the follower and following distribution of the users. The follower and following of the engager users in log-log scale shows a power law distribution (Figure 4.2). However we see a spike in the following distribution which based on previous work [Ali+19; Jia+14] can be an indication of bot accounts. On Twitter you can't choose who follows you but you can choose who you want to follow. The spike shows that there are some accounts that in an abnormal way choose to follow lot's of accounts.

4.2.3 Hashtag duration

Another observation from the dataset is the distribution of hashtags based on how long they remain alive. This can be defined as the hashtag duration which can be calculated by taking the difference between the last and first timestamp of when the hashtag was observed.



(a) followers **(b)** followings

Figure 4.2: The spike in the followings shows an abnormality and is an indication of bot accounts following lots of users.

Figure 4.3 shows the distribution of hashtags based on their duration in days and note the y-axis (frequency) is in log scale. First, we can see that a lot of hashtags have a life span of less than half a day. Second, in 24-hour cycles, there will be hashtags that become trending hence the spikes. Another conclusion we can make from this chart is part of the rich get richer phenomenon [DJ10] that hashtags which can make it to the top, will also remain alive for a longer period of time.

We can also see a similar pattern for the duration of the shared URL links.

4.3 Methodology

In the section we will describe the extracted features and the final model for predicting engagements. Figure 4.4 shows the overall architecture of our approach.

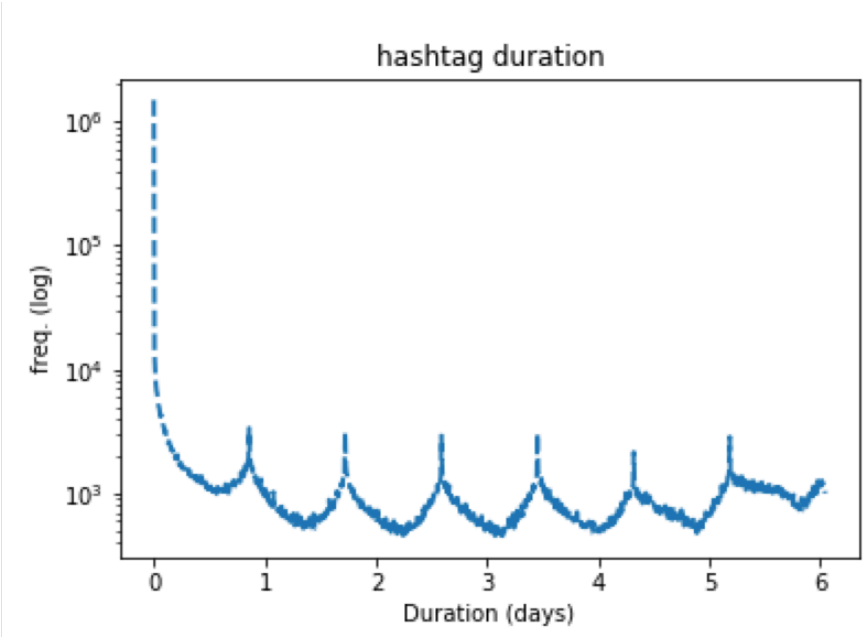


Figure 4.3: Some hashtags become trending each day. The trending hashtags also last for a longer duration.

4.3.1 Tweet features

The features extracted at the tweet level is considered based on unique tweets in the dataset. In other words, the features for tweets are considered regardless of the amount of engagements they received.

The tweet features consists of two main parts: the textual feature (tf_idf vector) and categorical, numerical or binary features. The final feature vector for the tweets has 32 elements were the first 16 are the tf_idf vector and the remaining features are hashed into a vector with 16 elements. Feature hashing is a method used to map from a high-dimensional space to specified dimension.

The categorical features regarding the tweet are as follows: tweet_type (TopLevel, Retweet, Quote), language, hour_of_tweet, number_of_BERT_tokens, has_hashtags, has_media, has_links, is_in_top_daily_hashtags, is_in_top_daily_links. The feature is_in_top_daily_hashtags is determined from Figure 4.3. We create this binary feature based on the hashtag duration and if the hash-

tag falls in the peak of a 24 hour cycle, we assign a true value to it. The is_in_top_daily_links feature is created in a similar manner.

4.3.2 User features

Following the same approach for the tweets, the features extracted for enagager and engagee users were filtered on each of the unique user sets.

The categorical features for the user consists of the account_year, is_verified and is_bot. We defined the is_bot feature based on the following count. If the user falls in the abnormal part of the following count distribution (see figure 4.2.b), we will consider this as a unique feature to give more weight to it on the final model. As for the numerical features, we compute 50 quantiles of the following and followers count. The categorical and numerical features are assembled and passed through a featurehasher with 16 elements.

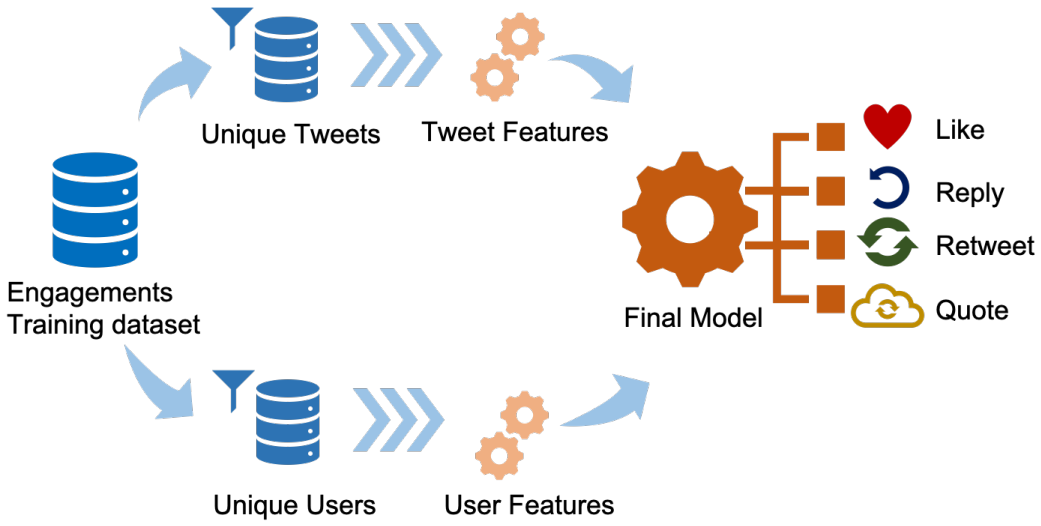


Figure 4.4: Architecture of the model.

4.3.3 Making Predictions

In the previous sections, we went through the preprocessing and feature engineering steps. In this section we would take the final feature representation as a feature vector to train four models for each of the four engagement types:

likes, replies, retweets and quotes. We classified each engagement type by using the gradient-boosted tree algorithm [Fri02]. We trained the model on a Spark cluster configured with 28-core 2.00Ghz with 200GB of RAM. With the default parameters of the GBTClassifier algorithm, each engagement took one hour to train. Further improvement on the results can be made by tuning the parameters of the GBTClassifier.

4.4 Evaluation Metrics

Since the final task is to predict whether a user will engage or not, the evaluation metrics for a classification problem can be used. More specifically the area under the precision-recall curve (PR-AUC) is a criteria that shows the tradeoff between the correct number of predictions and the overall amount of predictions. The PR curve is useful for applications where the focus is on the positive matches and especially when the positive set is relatively smaller than the negative set [Sta17]. This holds true for each engagement type (e.g. the number of tweets that get a quoting engagement compared to all the tweets).

Another metric is the Relative Cross Entropy (RCE) which evaluates the model by comparing it to a naive predictor that does not take any features into account and is based on the flow of engagements that enter the system (see Figure 4.1). The RCE can be calculated as follows:

$$RCE = (1 - \frac{\overline{CE_{pred}}}{\overline{CE_{naive}}}) * 100 \tag{4.1}$$

where the $\overline{CE_{pred}}$ and $\overline{CE_{naive}}$ are the average cross entropy of the model and the naive predictor respectively.

The final results of our method based on the final leaderboard of the competition is reported in table 4.2. Based on this table and figure 4.1 we can see predicting on engagements becomes more challenging as less data is available in the training dataset.

Engagement type	Area Under Precision-Recall Curve (PRAUC)	Relative Cross Entropy (RCE)
Like	0.5879	7.21
Retweet	0.1703	2.67
Reply	0.0593	-6.27
RT with comment	0.0099	-62.98

Table 4.2: Results based on the final leaderboard

4.5 Future work on the dataset

There is been a recent push toward standardizing and benchmarking graph datasets by the research community [Dwi+20]. The RecSys dataset can be added as part of the large social datasets for Open Graph Benchmark [Hu+20]. The dataset includes the follower information of the users and the retweet information of tweets. Therefore studying the user graph and rewteet graph in a standard format could be a possible future direction.

In section 2, we talked about the uniqueness of the ACM RecSys dataset. Another information that can be found in the dataset that can not be retrieved from the API is the retweet graph. From the API, it is not possible to collect the retweet sequence of users. In other words, we can not find the information on whom did the user retweet the original tweet. As this information is not available various approaches like the time-inferred diffusion model has been used instead of the true retweet graph [VRA18a]. Another future work can study the diffusion of tweets based on the true retweet graph.

4.6 Conclusion

In this chapter, we proposed our method for predicting user engagements for the ACM RecSys 2020 challenge. In particular we discussed the observations derived from the dataset and how they were implemented in the final model. We modeled the challenge as a binary classification problem and for each engagement type we trained a gradient boosting classifier. The dataset that was released as part of this challenge includes information that is not available

through the Twitter API. As for future work, we will focus on other aspects of the dataset namely the retweet graph.

5 Using GNNs for User Engagement Prediction

In this chapter, we introduce a novel approach for predicting user engagement on online social media using graph neural networks. While traditional methods rely solely on user and content features, our method also incorporates graph structure. We utilize pre-trained GloVe embeddings to generate features for tweet text and GraphSAGE for user embedding. Our GNN model combines these features with graph structure to predict user engagement. We apply our model on the same dataset introduced in the previous chapter for classifying tweet engagements as like, retweet, reply, or quote.

5.1 Introduction

In today's world of attention economy, predicting user engagement is a crucial part for online content generators. Many different approaches have been applied to predict user engagement. However these methods rely solely on features based on the user and the content. A new class of deep learning methods based on graphs captures not only the content features but the graph structure as well. In this chapter, we design a graph neural network to predict user engagements with tweets based on the features of the user, the tweet and the engagement graph of users and tweets. We first create features for the tweet text using the pre-trained GloVe embeddings and use GraphSAGE for embedding a user into a vector space. Then our GNN model combines the features and graph structure to predict the user engagement. Using a dataset officially released from Twitter, we are able to classify between like, retweet, reply and quote on a tweet. Our method is able to predict the final engagements with 94.22% precision score.

Social networks like Twitter, Facebook enable users to generate content and communicate with each other on their platform. As users tweet or post ideas and messages on their timelines, others can also interact with the content by liking, sharing, adding comments and showing their engagement with the topic. Understating how users decide to engage with various content is

an important research field in many domains like marketing, user behaviour analysis, online advertising and etc. Many business models are based on attracting users' attention on these social media platforms. Therefore predicting user engagements allows companies and individuals to increase their reachability and target their followers in more efficient way.

There are many challenges when it comes to predicting user engagements. First, collecting social network data has its limitations. Even though most platforms provide access to their content through APIs, there is a limit on the amount of data and type of data than one can retrieve. Furthermore some studies show the collected data can be biased towards the sampling technique of the API [Mor+21]. Second, previous approaches to the task of user engagement prediction do not take into account the network structure between users and the content i.e. tweets. Thus, classical methods which depend solely on user and tweet features are inefficient. Finally, engineering features in classical machine learning approaches is challenging because they require domain knowledge.

In this chapter, we create a graph neural network model to predict the engagement of users on tweets. We use a rich dataset of tweet engagements released by the ACM RecSys Challenge and Twitter [20]. We preprocess the dataset and create a heterogeneous graph of users and tweets. Our model takes the tweet features from pretrained GloVe embeddings and creates user embeddings for users. In the final step the GNN model aggregates the embeddings to predict the engagement of a user on a tweet.

We evaluated our model with different hyperparameters to find the optimal embedding size for the user and the tweets. Our proposed model shows a high performance in classifying user engagements.

5.2 Dataset

The 2020 and 2021 ACM Recsys challenges focused on predicting tweet engagement. As part of these challenges, Twitter released a unique dataset of public tweets and their engagements. The dataset not only provides the engagements (like, retweet, reply, quote) a user had with tweets but also the non-engagements. The non-engagements include tweets a user observed in his timeline but showed no interest in engagement. This is an important

part of the dataset and makes the dataset unique because such data is not available from the Twitter API. In other words, it is possible to collect tweets and engagements with tweet via Twitter API. However, the data of a user who has seen a tweet and implicitly opted out of an engagement cannot be retrieved from the Twitter API.

The ACM Recsys Challenge 2020 dataset consist of 160 million tweets over a span of 2 weeks. Figure 5.1 shows the flow of engagements in the first week of the dataset. In this paper, we have used a subsample from the first day of this dataset.

	Count
Unique users (engager and engagee)	1,140,443
Unique tweets	422,475
Engagements	858,140

Table 5.1: The sampled dataset statistics

Table 5.1 shows the summary of the number of unique users and tweet we use in this work. We have more than 1 million users engaging with more than 400 thousand tweets. If a user engages with one tweet, there will be an edge between the user and tweet. We will randomly split the dataset based on the number of edges for training and testing.

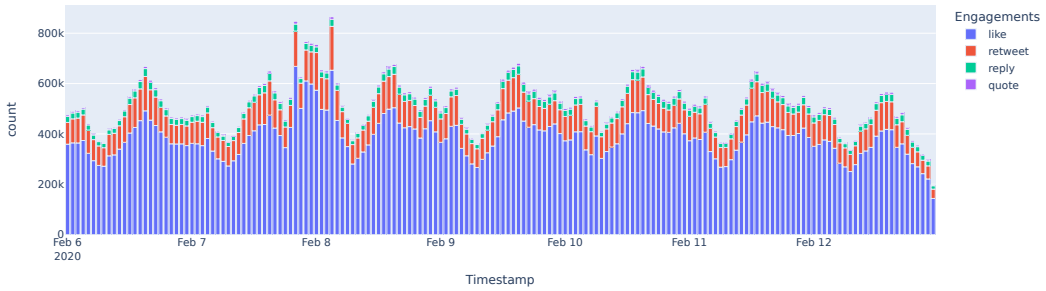


Figure 5.1: Engagements over time

5.2.1 Preprocessing the data

In order to prepare the data as an input for our model, we apply three main preprocessing steps on the tweet text and the engagement labels and create a grpah of users and tweet accordingly. First, we decode the text_tokens field from BERT embeddings [Dev+18] into text and transform it to GloVe embeddings [PSM14]. We use two of the pretrained GloVE models on 27 billion tweets to transform the text field of the tweets into tweet features. More specifically we use the 25 and 50 dimensions pretrained models. The GloVe model creates word embeddings based on the co-occurrence of words within text data. The tweet features is represented based on the average GloVe embeddings of each word in the tweet. We chose GloVe over BERT embeddings because we wanted a fixed size of vectors for all tweets.

In the second preprocessing step, we created new labels based on the combinations of the possible engagements with a tweet. There are four types of engagements (like, retweet, reply and quote) available with a tweet on Twitter. In order to create one model for predicting the final engagements, we create the engagement labels as follows:

As we can see in Table 5.2, we have 12 labels based on the four main engagements types. The distribution of the labels are imbalanced with almost half of the tweets showing no engagements. We will use a weighting mechanism to tackle the imbalance nature of the data.

As the final step in the preprocessing phase, we create a heterogeneous graph of users and tweets as nodes and engagements as edges. Each edge has a label based on Table 5.2.

5.3 Methodology

5.3.1 Problem definition

Given a user $u_i \in U$ and a tweet $t_j \in T$, predict the type of engagement label y_{ij} of user u_i on tweet t_j. Since we create a graph of engagements between users and tweets. We formulate this classification task as a link prediction problem where the engagement label will be the weight on the edge between the user and tweet to be predicted.

Engagement type	Label	Distribution. (in percent)
no-engagement	"0"	48.91
like	"1"	37.28
like & retweet	"2"	5.29
retweet	"3"	5.03
reply	"4"	1.62
like & reply	"5"	0.88
retweet & quote	"6"	0.44
like & retweet & quote	"7"	0.3
like & retweet & reply	"8"	0.15
retweet & reply	"9"	0.06
retweet & reply & quote	"10"	0.02
like & retweet & reply & quote	"11"	0.02

Table 5.2: Modeling the labels with different engagements types

5.3.2 Graph Neural Networks (GNN)

Graph Neural Networks are the new class of neural networks which are able to take graph structures into their learning architecture. These networks can learn embeddings based on the nodes' features and the features of their neighborhoods. Kipf et al. [KW16] proposed graph convolutional network (GCN). Their approach requires the full graph Laplacian to be calculated and the output embeddings of a node in each layer is dependent on all it's neighbors at the previous layer. Most recently Hamilton et al. [HYL17b] introduced GraphSage. In their method, they tackle the problem of having to deal with the entire graph Laplacian and show how to aggregate information from a node's neighborhood. The main idea of aggregating features and passing it through a non-linearity has been applied in many different problems[Ali+19; HYL17b].

5.3.3 Proposed Model

Our model uses GraphSAGE [HYL17b] as the graph convolutional layer. First, we pass the user through an embedding layer to get its embedding vector. We concatenate the user embedding vector with the tweet vector and then pass it through the SAGEConv layer based on the graph structure. Finally, the output of the convolutional layers is passed through a linear layer to predict the final engagement based on the edge label types.

Each user u_i has an embedding vector h^{u_i} which can be accessed through the embedding layer weights:

$$h^{u_i} = Z^T \cdot u_i. \tag{5.1}$$

Where $Z \in R^{|U| \times d}$ is embedding layer. The dimension of the embedding layer is based on the number of all users and embedding size d which can be configured as a hyperparameter.

As explained in the preprocessing phase, for each tweet, we have a fixed size of the GloVe embeddings h^{t_j}.

The output of the SAGEConv layer h_v^k at each depth is calculated as follows:

$$h_{N(v)}^k = mean(\{h_u^{k-1}, \forall u \in N(v)\}). \tag{5.2}$$

$$h_v^k = \sigma(W^k \cdot g(h_v^{k-1}, h_{N(v)}^k)). \tag{5.3}$$

Where $h_{N(v)}^k$ is the average of the embedding vectors from v's neighbors. h_v^k is the output which is concatenated with v's previous embedding. σ is the non-linearity function which in this case we use the ReLU function. The function $g(.)$ is the aggregation function collecting embeddings from the nodes neighborhood. We use the mean function for the aggregation function.

In the final step our model makes predictions $\hat{y}_i$ by passing the outputs of the SAGEConv layer to a linear layer for the engagement labels. Our model learns the weight of the links based on the training data. We define the loss function as the weighted mean square error of the prediction and actual numerical value of the engagement type.

As we have formulated this problem into a link prediction task, our model learns the weight of the links based on the training data. We define the loss

function as the weighted mean square error of the prediction and actual numerical value of the engagement type.

The output of our model will be the numerical value representing the engagement label. We convert the predicted numerical value to the closet integer as a string to represent the engagement label. We are now able to evaluate our model in terms of the precision, recall, and F1-score metrics.

As we have formulated this problem into a link prediction task, our model learns the weight of the links based on the training data. We define the loss function as the weighted mean squared error of the prediction and actual numerical value of the engagement type:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} w_{y_i} * (y_i - \hat{y}_i)^2. \tag{5.4}$$

where w_{y_i} is the weight assigned to engagement type label and is calculated as follows:

$$w_{y_i} = max(w_y)/count(y_i). \tag{5.5}$$

This simple weighting mechanism, penalises the model based on the imbalanced distributions of the labels.

5.3.4 Implementation

We implemented our model using the Pytorch Geometric library [22a]. We choose the Adam [KB15] optimizer with a learning rate of 0.01 and trained the model with 600 epochs in each run. We split the dataset into 90% training and 10% testing data.

5.4 Evaluation

As we mentioned in the preprocssing step, we used GloVe embeddings to represent the tweets. Our model also creates embeddings for the users. In this section we evalute the different embeddings size we experimented with our model.

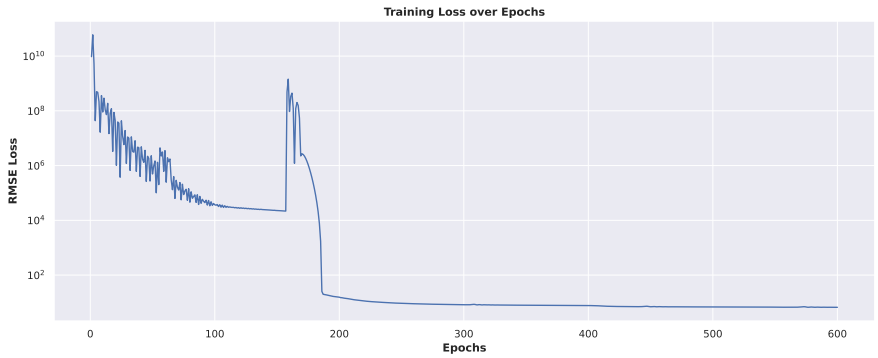


Figure 5.2: Training loss

Figure 5.2 shows the training loss over the 600 epochs. We see the convergence after 200 epochs.

In Table 5.3 we see the final loss values for different embedding sizes. The best trained model is the when the embedding sizes for users and tweets are both 25. As we increase the embedding sizes, we are adding more parameters to the model and therefore training the model becomes more difficult. As we mentioned in the previous section we use a weighted loss function. Here we report the weighted and non-weighted loss on training data. We can see that training based on the weighted loss, helps achieve a better loss score for the final model. We can also see that considering a bigger embedding size for the users is more beneficial for the model than using a bigger embedding size for the tweets.

User-GloVE embedding size	Weighted train	Train	Test
	MSE		
25-25	6.635	0.441	2.748
25-50	44.131	3.029	491.432
50-25	654.904	15.992	20.317
50-50	967.832	25.274	806.919

Table 5.3: MSE Results with different embedding sizes

We use Precision, Recall and the F1 score for the final evaluation of the model. These metrics make sense since we are classifying the engagements. The Precision shows the fraction of engagements that where classified correctly while Recall calculates the classified engagements over all the relevant engagements of tweets and users. The F1 score is the harmonic mean of the Precision and Recall scores. Table 5.4 shows the results of the classification metrics. We can see by using an embedding size of 25 for both users and tweets we can reach a 94.22% Precision and 92.16% Recall with F1 score of 94.16%.

User-GloVE embedding size	Precision	Recall	F1
25-25	94.22%	94.16%	94.16%
25-50	89.27%	88.70%	88.41%
50-25	90.35%	90.12%	89.79%
50-50	90.41%	90.15%	89.60%

Table 5.4: The Precision, Recall and F1 score on different embedding sizes

5.5 Related work

There has been many studies focusing on user engagement in online social networks [20; ABM20; Bel+20]. As their final solution for perdicting user engagements, Schifferer et al. [Sch+20] proposed three independent XGBoost models trained on different features and different subset of the dataset. They created a data pipeline that runs completely on GPU. This allowed experimenting various combinations of features and tuning the hyperparameters in a fast and efficient way. Alhosseini et al. [ABM20] created separate binary classifiers for each engagement type using gradient boosting.

Ma et al. [Ma+21] use GNNs on Heterogeneous Information Networks (HINs) to predict customer engagements on Facebook posts for company brands. Their model makes use of two interesting techniques. They traverse through specific paths on the graphs creating meta-paths to capture the

interactions between users and posts. They also add a attention mechanism to help interpret on the final results of the their model.

5.6 Conclusion

In this chapter, we used graph neural networks to predict user engagements on Twitter. We created a graph of user-tweet engagements with the user and tweets as the nodes and the engagement type as the edges. Using GNNs enabled us to make predictions based on the user, tweet and the engagement graph. Our model used the pretrained GloVe embeddings for the tweet features and created user embeddings to predict the final engagement type for a user and tweet.

In future work, we would investigate expanding on the heterogeneous graph of users and tweets by adding other entities like hashtags, tweet hour and etc. This could be achieved by defining a GCN layer to work on multiple edge types. Another avenue of investigation is to train the model based the negative log likelihood of the label classes. This could give more detail on how far each engagements is from one another.

6 A GNN Approach for Bot Detection

In this chapter, we take a look at another application of graph neural networks on classifying accounts on social media based on the spam bot behavior. GNNs can be applied for spam bot detection since they can incorporate both the feature vector of accounts and the graph structure between the accounts.

6.1 Introduction

Spam Bots have become a threat to online social networks with their malicious behavior, posting misinformation messages and influencing online platforms to fulfill their motives. As spam bots have become more advanced over time, creating algorithms to identify bots remains an open challenge. In this paper, we propose a model based on graph convolutional neural networks (GCNN) for spam bot detection. Our hypothesis is that to better detect spam bots, in addition to defining a features set, the social graph must also be taken into consideration. GCNNs are able to leverage both the features of a node and aggregate the features of a node's neighborhood. We compared our approach, with two methods that work solely on a feature set and on the structure of the graph. To our knowledge, this work is the first attempt of using graph convolutional neural networks in spam bot detection.

With the advent of Online Social Networks (OSN), the way people share and receive information has changed. Today, users tweet and retweet on Twitter, post and share messages on Facebook and upload videos on YouTube in the scale of millions. Most of these actions are carried out by users Vosoughi et al. [VRA18b], however automated computer programs named *Bots* are influencing users in the content their sharing.

Social media websites like Twitter were created on the premise of users generating content on their platform. In 2018, the amount of tweets and posts shared every day is reported to be in the scale of hundreds of millions Chu et al. [Chu+10]. However, with every technology comes abuse and those who would want to take advantage of it. Users are not the only ones

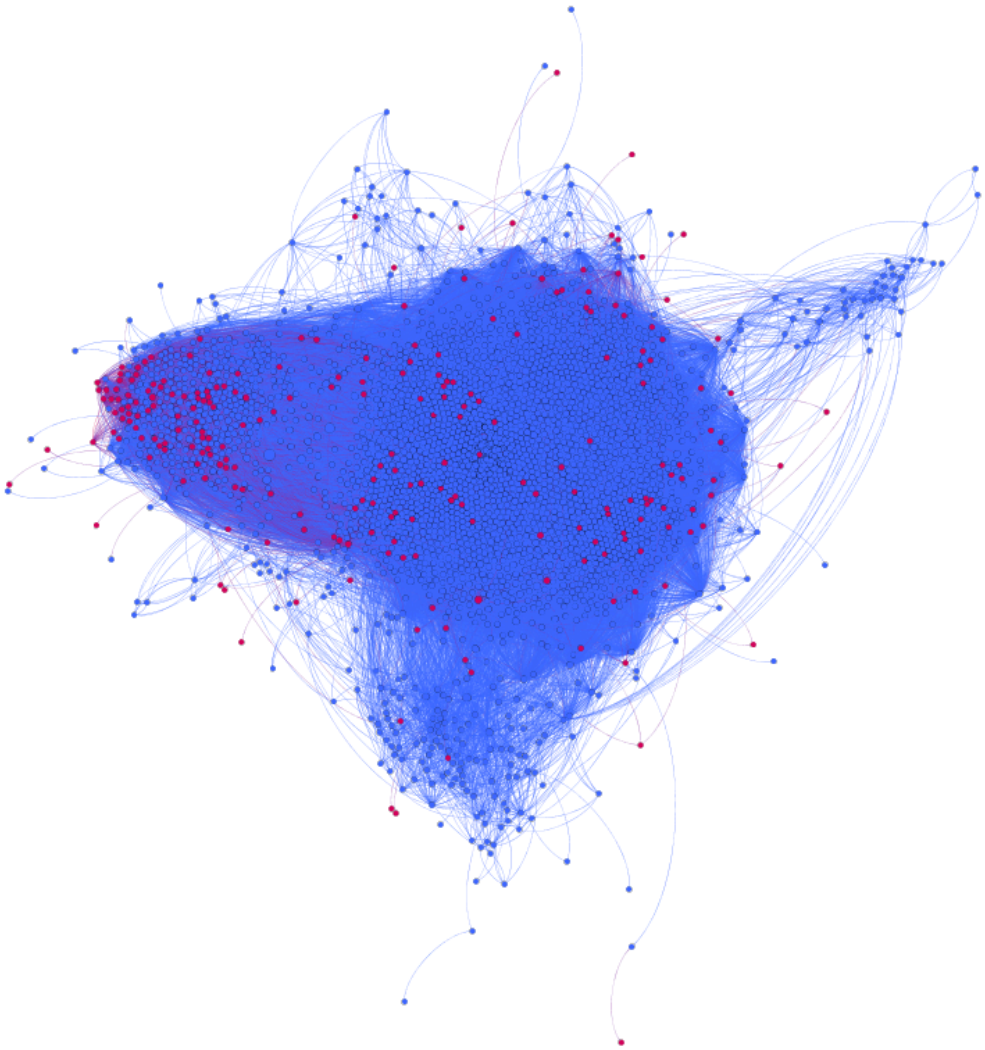


Figure 6.1: Bot and User in graph structure. Red nodes indicate bot accounts while the blue nodes show the user accounts

using social media platforms. In fact, Varol et al. [Var+17] reported around 15% of tweets on Twitter are published by automated accounts named bots. Bots have influenced social networks by promoting campaigns, spreading misinformation and selling accounts when they get popular. Twitter has tried to suspend such accounts but there are more bot accounts created than suspended Chavoshi et al. [CHM17]. Various research groups have investigated the problem of bot detection using different techniques and approaches. The difference in these work varies depending on the definition of a bot account, the selected feature set representing accounts and the machine-learning algorithm used for classifying bot accounts from normal user accounts. Despite the detection algorithms introduced by the research community, the problem of bot detection remains an open problem. As reported by Chavoshi et al. [CHM17], Cresci et al. [Cre+17], and Yang et al. [YHG13], bots have become more advanced and sophisticated in avoiding the existing proposed detection methods. In fact, bots have been evolving over time. For instance, early bots were detected by posting similar tweets. Soon bots started using alternative words and synonyms to avoid posting similar content. Ferrara et al. [Fer+16] has given attention to the rise of social bots that are designed to emulate human-like behavior. The social bots are able to interact with other accounts, post tweets in different topics, and display a similar activity like humans Cresci et al. [Cre+17]. Therefore it is essential for bot detection methods to consider the characteristics of groups of accounts, rather than merely focus on each account individually.

The main contributions of this chapter are summarized as follows:

1. This is the first attempt to use the graph structure of Twitter accounts in the learning phase for detecting spambots.
2. We deploy graph convolutional neural networks on a well-known dataset previously used in the literature.
3. We show that using the graph structure in our method gains better performance in spambot detection.

The remainder of this chapter is structured as follows. First we cover the previous related work in spam bot detection and graph convolutional neural networks. Section 6.2 describes in detail the dataset used. In section 6.4, we provide an overview of our methodology. We illustrate are results in section

6.5 and discuss the limitations of our work and suggestions for future work. Finally, we conclude this chapter in section 6.6.

6.2 Related Work

In this section, we first review the literature on spambot detection and compare each work by their definition for spambots, the features used and the classification algorithm they employed. Next, we look at graph convolutional networks.

Lee and et al. [LEC11] proposed a method working as a honeypot trap for bot accounts. They created 60 twitter accounts and started posting meaningless tweets that would have no interest for humans. Despite this fact, they were able to draw some accounts' attention to follow the accounts they made. Analyzing these accounts in detail showed that they were in fact bot accounts trying to increase their following list.

Yang and et al. [Yan+12; YHG13] used a conservative definition for bot accounts considering only accounts who post URLs linking to malicious content. They also introduced and considered several robust features on the BayesNet classifier to predict spam accounts. Yang and et al investigated the different approaches bots take for avoiding detection by Twitter. Their findings show that bots tend to increase the reputation of the their accounts by purchasing followers and posting more tweets.

In [Cre+16] the authors introduced a DNA-inspired technique that models each account as a sequence of behavioral information and detects spambots based on similar sequences. They categorized each users' tweets into different types and based on whether a tweet contains URLs, hashtags, pictures, etc. it will be assigned a different character. The similarity of the accounts is measured by the longest common substring in their DNA sequences.

BotOrNot [Dav+16] used the random forest classifier algorithm on more than 1000 features to detect bots. The features are categories in 6 groups: network (degree distribution, clustering coefficient, ...), users' account information, friends (number of follower, followings, ...), temporal (tweet rate, ...), content (natural language processing, ...) and sentiment features. The downside of BotOrNot is that it was trained on English tweets so its

performance declines on bots which are tweeting in another language than English.

DeBot [CHM17] developed by Chavoshi and et al. is an unsupervised bot detection system. The idea behind their work is that accounts with a high correlation in their activities (tweet, retweet, ...) have a high chance of being bots. DeBot monitors the activities of accounts over a specific period and creates a time series for each account. It then clusters accounts based on the similarity of their time series using a lag-sensitive hashing method. Finally, DeBot reports the accounts with a high correlation as bots.

[Nas+18] defined spam bots as content polluters that try to takeover a discussion for political or advertising reasons. Their approach considers individual tweets for detecting bots. Instead of focusing on the friend and follower network, they created the event network where the nodes are the users and the edges are based on users having tweeted on the same event. They also compute the diversity of a tweet based on the URLs and hashtags it has mentioned. Results of their work indicate that spam bots operate as a group often tweeting at the same time.

6.2.1 Graph convolutional networks

Graph structures are used in many domains and applications such as social networks, recommender systems etc. The challenging task for graph structures is how to use them in machine learning algorithms. The initial works in this area considered the statistical data of the graph like the degree of the nodes, the centrality and betweeness coefficients as features for training models. In other words, they considered the graph structure as a pre-processing step to extract structural information. Therefore, these approaches do not use the graph structure in the learning phase. Another downgrade for these approaches is that computing the graph statistics has a high complexity and the output of it cannot be used on unseen data.

Kipf and Welling [KW16] proposed graph convolutional network (GCN); a convolutional neural network approach on graph structures. The term convolutional is used since a node's neighborhood is considered as its representation. Their method can be considered as the initial steps for graph semi-supervised classification tasks. However, the drawbacks of their approach is that it requires the full graph Laplacian to be calculated and the

output embeddings of a node in each layer is dependent on all it's neighbors at the previous layer.

Most recently [HYL17a] introduced GraphSage; a node embedding algorithm that uses neural networks to learn embeddings for nodes in the graph structure. Their main contribution is that they solve the limitations mentioned above and show how to aggregate information from a node's neighborhood. Their method consists of two main phases:

1. **Defining the computation graph and training the neural nets**

   The structure of a node's neighborhood will define the computation graph for training the neural networks. In this phase, the objective is to build neural networks that will ensure nodes close to each other have similar embeddings while nodes far from one another have different embeddings.

2. **Propagation** For each node the information of its neighbors is aggregated and passed through the neural networks trained in the first phase.

6.3 Dataset

There are several well-known datasets collected by different research groups specifically for bot detection on Twitter. Lee et al. [LEC11] provide a social honeypot dataset that contains approximately 22000 content polluters. They have gathered the accounts' metadata and tweets of each account. However in their released dataset they have anonymized the Twitter account ids. Therefore collecting further information is not possible. Cresci et al. have worked on different Twitter datasets in [Cre+17] and by using a crowdsourcing platform they labeled the different types of accounts. [Var+17] released the twitter ids of the accounts they detected as spambots.

Yang et al. [YHG13] collected Twitter spammers and their dataset contains each account's followers and followings. To the best of our knowledge, this is the dataset we found which has gathered this information for the Twitter accounts. The authors of that work have kindly shared their dataset and we have used the dataset in this work. The dataset contains 11000 nodes and 2342816 edges between them.

Table 6.1 shows the statistics of dataset used in this paper. The age, tweets and neighbors columns indicates the average amount in each group. The age column is the average age of the accounts reported in days. The majory of edges between nodes are user to user connections. However around 5.4% of the edge relations include bot accounts.

	Accounts	Age	Tweets	Neighbors
bots	1000	3023.80	220.90	1963.84
users	10000	3174.28	4658.52	21579.76
relation	bot-bot	bot-user	user-bot	user-user
	2673	73363	50153	2216627
	0.11%	3.13%	2.14%	94.61%

Table 6.1: Dataset statistics

Figure 6.2 shows the degree distribution of the accounts in the dataset. Most accounts have a small number of followers and followings and there are a few accounts which have more than 1000 accounts in their neighborhood.

Figure 6.3.a shows the age and the length of user account name for both bots and users accounts. As shown in figure 6.3.b and reported in previous work [Fer+16] bot accounts have smaller age meaning they were created more recently compared to user accounts. Also as [Nas+18] indicated there is no significant difference in length of the accounts name.

6.4 Methodology

We used an inductive representation learning approach similar to [HYL17a; HYL17c] for detecting twitter bot accounts.

Problem definition. Let $G = (V, E)$ be a graph where for each $v \in V$ exists a feature vector X_v and a binary label $y \in \{0, 1\}$ associated with it. The goal is to find an embedding vector h_v for each node $v \in V$ such that $f(h_v)$ predicts the label of the node in the graph.

Similar to convolution filters in image processing, graph convolutional networks consider the attributes of a node's neighbors as a representation

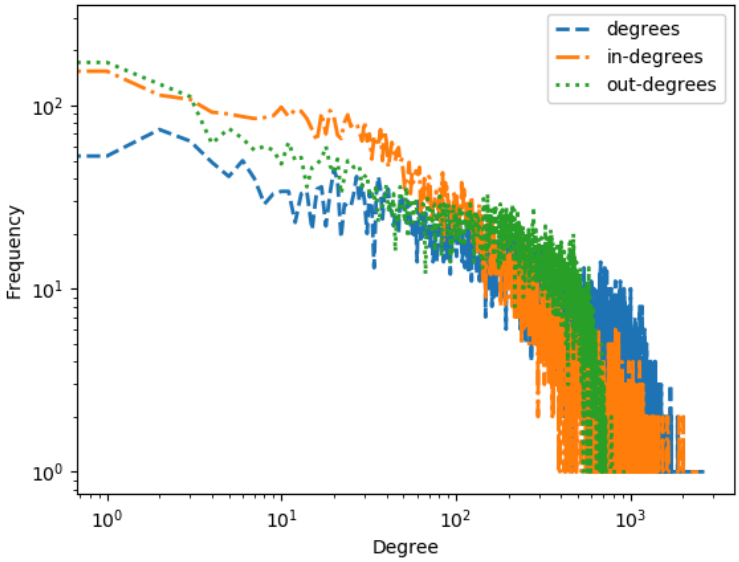
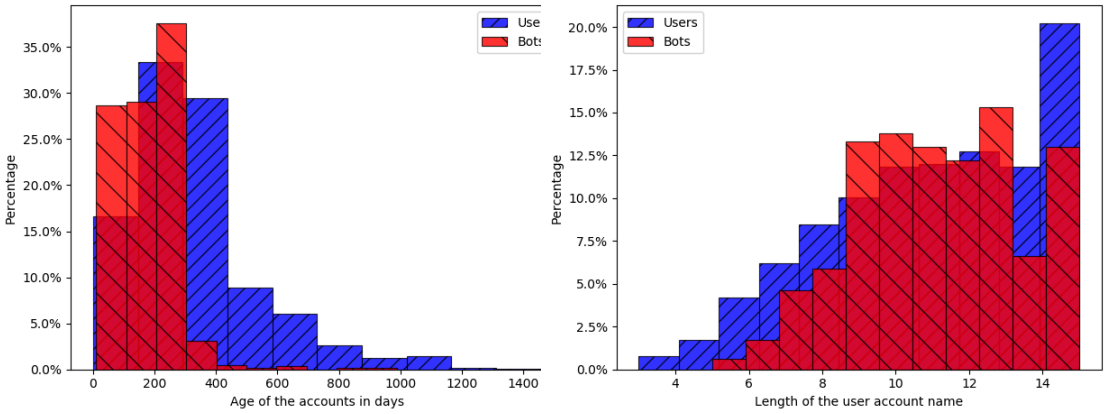


Figure 6.2: The degree distribution of the nodes in graph. The figure is drawn in log-log scale.



Age (a) Account Length Name (b)

Figure 6.3: Bots(red) and Users(blue) attributes

for that node. Let us define k as the depth of the neighbors of a node from which information is aggregated. If k=1 only the information from its own neighbors will be considered. For k=2 the information is gathered also from the neighbors of its neighbors and so on. The output h_v^k at each depth is calculated as follows:

$$h_{N(v)}^k = mean(\{h_u^{k-1}, \forall u \in N(v)\}) \tag{6.1}$$

$$h_v^k = f^k(h_v^{k-1}, h_{N(v)}^k) = \sigma(W^k \cdot concat(h_v^{k-1}, h_{N(v)}^k)) \tag{6.2}$$

Where $h_{N(v)}^k$ is the average of the embedding vectors from v's neighbors. h_v^k is the output which is concatenated with v's previous embedding.

The neural networks are optimized based on the cross-entropy loss function:

$$J(f^k(h_v^{k-1}, h_{N(v)}^k), y) = -\sum_{v \in V} y \log(f(X_v)) + (1-y)\log(1 - f(X_v)) \tag{6.3}$$

6.4.1 Features

The initial vector (X_v) for each user consists of the features that can be retrieved directly from the Twitter API (Table 6.2). The feature vector consists of the created date of the account, the number of favourites, followers, followings and statuses of the account. We considered the length of an account's name and computed the created_at in days as features.

6.5 Evaluation

We conducted a 5-fold cross-validation on the dataset to evaluate the accuracy of the model. Figure 6.4 shows the area under curve for each fold. On average the GCNN has 0.94 accuracy measured by the area under curve.

We used the precision, recall and f1 metrics as shown in Table 6.3 for the evaluation. Choosing a meaningful evaluation metric for the classification task is important. For example, it is possible to use the precision measure

Feature Name	Description
Age	The created_at attribute returns the datetime that an account was created on Twitter. The age feature is computed by the number of days from the created_at date.
favourites_count	This feature indicates the number of tweets a user has liked.
followers_count	followers_count shows the number of follower an accounts has.
friends_count	The friends_count attribute shows the number of accounts the user is following.
statuses_count	The number of tweets including the retweets a user has posted.
Account length name	The length of an account's name

Table 6.2: Features

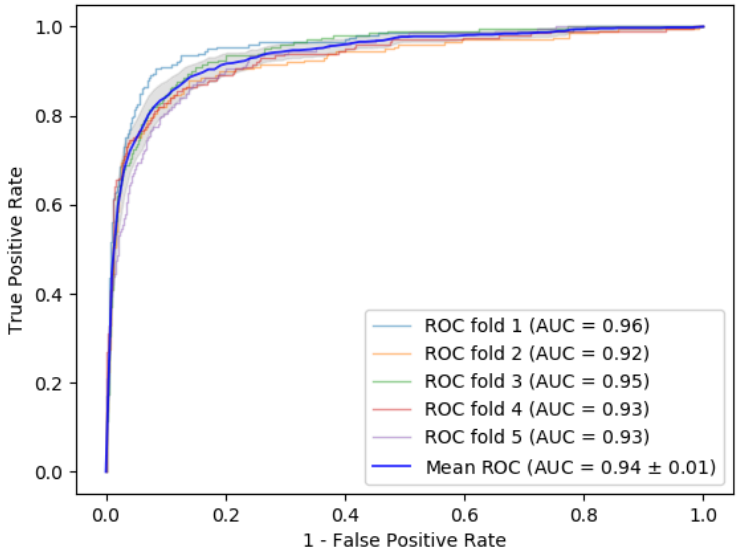


Figure 6.4: ROC curve over 5-fold cross-validation

defined in equation 6.4 to evaluate the performance of a model.

$$Precision_{micro} = \frac{\sum_c TP}{\sum_c TP + \sum_c FP} \tag{6.4}$$

However, by this definition for a dataset where the majority of labels belong to one class, the precision score remains high even if the model has not detected the labels of the other class correctly. Therefore, for a better evaluation of the model we compute the precision, recall, f1 score for each class separately and report the average score on the two classes. This is also known as f1 macro score in the *scikit-learn* python library.

$$Precision_{macro} = \frac{1}{|c|} \sum_c \frac{TP}{TP + FP} \tag{6.5}$$

We plotted the Receiver Operating Characteristic (ROC) for the different models as shown in figure 6.5. We observe that the area under the ROC curve is 94% for the GCNN approach which is 8% and 16% percent higher than the MLP and BP approach respectively.

Since the Twitter API has a limit of 15 requests every 15 minutes, building the Twitter graph structure based on the follower and friend relation of the accounts is not an easy task. We are aware this may be considered as a limitation to our approach. It can thus be suggested to build the graph structure based on retweet graph of user accounts.

6.5.1 Comparsion with MLP and Belief Propagation

We further evaluated our approach by comparing it with two other methods. As graph convolutional neural networks take both the feature set and the graph structure into consideration, we demonstrate the performance of this method by comparing it with multi layer perceptron (MLP) and belief propagation (BP). The MLP method is trained based on the feature set defined in section 6.4.1. On the other hand, the belief propagation algorithm runs solely on the graph structure.

The belief propagation algorithm [Pea14] labels each node in the graph based on the local observation of that node and the messages passed to it from other nodes. The message sent from one node to another indicates

	$f1_{macro}$	$Precision_{macro}$	$Recall_{macro}$
MLP	0.77	0.81	0.73
BP	0.55	0.56	0.54
GCNN	0.79	0.87	0.73

Table 6.3: Comparison of different algorithms on the dataset

its belief about the state of the other node. In other words, the algorithm works solely on the graph structure and propagating the labels of the nodes throughout the graph.

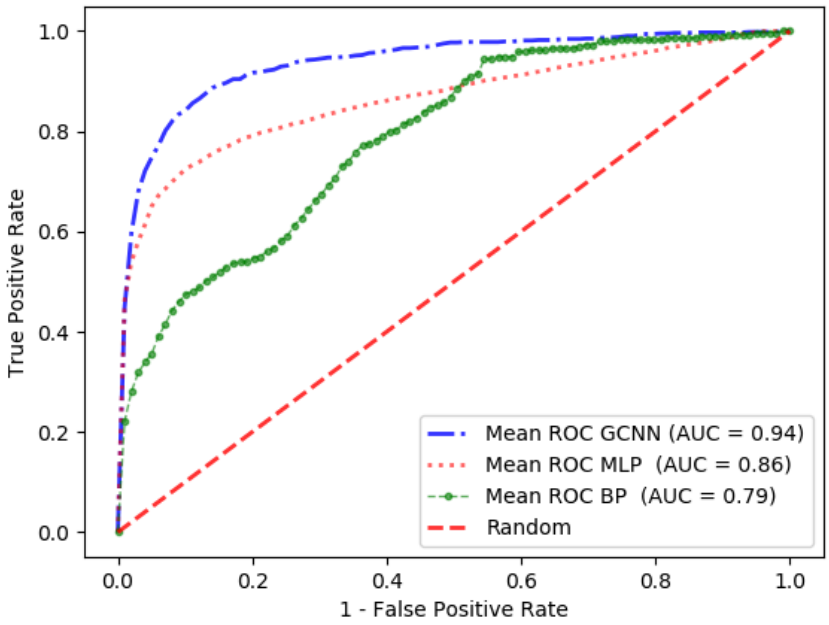


Figure 6.5: Comparison of the area under curve of different algorithms

6.6 Conclusion and Future Work

In this chapter, we have examined a new approach for detecting malicious accounts and social bots on Twitter by using graph convolutional networks. The main idea of our method is to employ the graph structure and relationships of Twitter accounts for classifying the accounts. Each account aggregates the feature information from its neighborhood. To demonstrate the efficacy of our proposal, we have worked on a previous well-known dataset in bot detection. Results show that our approach outperforms the state of the art classification algorithms with 8% improvement in the area under curve accuracy. Finally, a specific extension for future work is to deploy this method in real time on Twitter's streaming API for spambot detection.

7 Conclusions

In this thesis, we apply graph neural networks (GNNs) on different classification, prediction tasks and evaluate our methods using online social media data. In particular, we develop a system for data collection from the Twitter API based on the tweet ID. In order to predict user engagements with tweets we use GNNs to create a graph of user-tweet engagements, where users and tweets are the nodes and engagement types are the edges. We also propose a new approach for detecting malicious accounts and social bots on Twitter by using GNNs, taking into account the graph structure and relationships of Twitter accounts. The system we develop for data collection from the Twitter API is based on the tweet ID and enables us to collect data for our different classification and prediction tasks. Our evaluations using online social media data show promising results for both user engagement prediction and bot detection.

In chapter 3, we studied the Snowflake algorithm of Twitter which is responsible for generating unique tweet IDs. The study revealed that the Twitter API is only capable of providing a limited sample of tweets. However, we demonstrated that by using the binary representation of tweet IDs, combined with access to multiple API keys, this limitation can be circumvented. The study also showed that it is feasible and practical to collect massive numbers of tweets through informed Snowflake prediction. The system developed for this purpose is not only easily deployable and scalable but also operates reliably. Instead of sampling the space, the sampled stream was used to estimate the distributions which proved to be a faster, more consistent, and more reliable alternative. Additionally, the estimation of necessary tokens for particular percentages of the Twitter stream for 2022 was updated.

Although a 99% Firehose stream recreation requires a large number of user tokens, and is likely to be noticed by Twitter, the framework developed in this study can still be used effectively by researchers from all fields to analyze a specific event within a certain time. It is difficult to analyze events that occurred more than a week ago due to the limited access to substantial

numbers of tweets which are only available through Twitter-selected data dumps or live APIs. However, with the proposed method and an adequate number of tokens, a 24-hour window of tweets can be collected over a longer period of time, providing more tweets than the sampled stream. It is important to note that the sampled stream is not representative of the Firehose due to bad actors that target this time slice, leading to a slight overestimation of the expected hit rate of Snowflakes. Nevertheless, this should not affect the general distribution of servers.

Chapter 4 of this thesis presents a novel approach to predicting user engagement for the ACM RecSys 2020 challenge. In this chapter, a detailed account of the observations derived from the dataset and their corresponding implementation in the final model is provided. The challenge was posed as a binary classification task, and a classifier with gradient boosting was trained for each interaction type. It is worth noting that the dataset provided in the above challenge contains information that cannot be retrieved via the Twitter API. The developed algorithm leverages the information available in the provided dataset and uses a binary classification framework to distinguish between different types of user interactions. Specifically, the approach used trains separate classifiers to predict four types of user interactions, including retweets, replies, quotes, and likes. We evaluated the proposed model by the area under precision-recall curve (PRAUC) and relative cross entropy (RCE) metrics.

In Chapter 5, we presented our use of graph neural networks (GNNs) to predict user engagements on Twitter. Specifically, we constructed a graph of user-tweet engagements, where the nodes corresponded to users and tweets, and the edges represented the engagement type. Through the use of GNNs, we were able to perform predictions based on the characteristics of the user, tweet, and engagement graph. Our model utilized pre-trained GloVe embeddings for tweet features and learned user embeddings to predict the final engagement type for a user and tweet. As part of future work, we propose investigating the extension of the heterogeneous graph of users and tweets by the inclusion of other entities, such as hashtags and tweet hour. This extension could be achieved by defining a graph convolutional network (GCN) layer to handle multiple edge types. Additionally, one can explore the training of the model based on the negative log likelihood of the label classes,

as this approach may provide further insight into the degree of separation between different engagement types.

Chapter 6 of our work proposes a novel approach for identifying social bots and malicious accounts on Twitter by leveraging graph convolutional networks. Our method aims to take advantage of the graph structure and the relationships between accounts for classifying accounts. Specifically, each account accumulates feature information from its neighbors to make a classification decision. To demonstrate the effectiveness of our approach, we conduct experiments on a widely used dataset in bot detection. Our results demonstrate that our proposed method outperforms the state-of-the-art classification algorithms, achieving an 8% improvement in the area under the curve accuracy.

7.1 Future work

Graph Neural Networks (GNNs) have emerged as a leading machine learning architecture for supervised learning with graph and relational input. GNNs have been shown to be effective in predicting user engagements on Twitter. However, as social media platforms continue to evolve and generate large amounts of heterogeneous data, the performance of GNNs in predicting user engagements needs to be further improved. To this end, exploring the use of attention mechanisms and meta-path based Heterogeneous Information Networks (HINs) to better exploit large-scale content consumption information. By designing a GNN model that incorporates attention mechanisms and meta-path based HINs, it is possible to learn more accurate structural feature representations of users and better predict likes, retweets, replies, and quotes on Twitter. As the field of GNNs continues to advance, there is significant potential for these models to make even more accurate predictions and improve our understanding of user behavior on social media platforms.

Detecting spam bot accounts on Twitter remains a challenging task as new forms of social spam bots emerge that closely mimic the characteristics of real users. As a result, the use of Graph Neural Networks has shown promising results in detecting these sophisticated spambots. However, there is still room for improvement in GNN-based methods to better detect social spambots. With the exponential increase of graph data, the scalability of GNNs remains

a crucial issue. However, current research proposes tackling this obstacle by implementing sampling paradigms, such as node-wise, layer-wise, and graph-wise sampling. As GNNs improve, they can better detect spam bot accounts on Twitter by learning the structural feature representations of users and identifying anomalous behaviors. By analyzing collective behaviors and groups as a whole, novel analytic tools can be developed to turn the tide in the arms race against such sophisticated spambots. Future work in this field could focus on implementing and improving these sampling techniques to allow GNNs to better scale to larger graphs and more accurately detect spam bot accounts on social media.

Bibliography

[20] *Twitter RecSys Challenge 2020*. http://www.recsyschallenge.com/2020/. [Online; accessed 30-June-2020]. 2020 (see pages 31, 42, 49).

[22a] *PyG (PyTorch Geometric)*. https://pytorch-geometric.readthedocs.io/. [Online; accessed 25-June-2022]. 2022 (see page 47).

[22b] *Twitter - Nutzerzahlen von Twitter weltweit 2024 | Statista*. https://de.statista.com/statistik/daten/studie/318483/umfrage/twitter-nutzerzahlen-weltweit-prognose. [Online; accessed 10-Februray-2022]. 2022 (see page 13).

[ABM20] Seyed Ali Alhosseini, Raad Bin Tareaf, and Christoph Meinel. **Engaging with Tweets: The Missing Dataset On Social Media**. In: *Proceedings of the Recommender Systems Challenge 2020*. RecSysChallenge '20. Virtual Event, Brazil: Association for Computing Machinery, 2020, 34–37. ISBN: 9781450388351. DOI: 10.1145/3415959.3415999. URL: https://doi.org/10.1145/3415959.3415999 (see pages 4, 49).

[Ali+19] Seyed Ali Alhosseini, Raad Bin Tareaf, Pejman Najafi, and Christoph Meinel. **Detect Me If You Can: Spam Bot Detection Using Inductive Representation Learning**. In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. San Francisco, USA: Association for Computing Machinery, 2019, 148–153. ISBN: 9781450366755. DOI: 10.1145/3308560.3316504. URL: https://doi.org/10.1145/3308560.3316504 (see pages 4, 34, 45).

[AM22] Seyed Ali Alhosseini and Christoph Meinel. **Predicting User Engagements Using Graph Neural Networks on Online Social Networks**. In: *IThe 8th IEEE International Conference on Data Science and Systems, DSS 2022, Chengdu, China, December 18-21, 2022*. IEEE, 2022, 200–205 (see page 4).

[AM23] Seyed Ali Alhosseini and Christoph Meinel. **The More The Merrier: Reconstruction of Twitter Firehose**. In: *International Conference on Information Networking, ICOIN 2023, Bangkok, Thailand, January 11-14, 2023*. IEEE, 2023, 200–205. DOI: 10.1109/ICOIN56518.2023.10048898. URL: https://doi.org/10.1109/ICOIN56518.2023.10048898 (see page 4).

[Bel+20] Luca Belli, Sofia Ira Ktena, Alykhan Tejani, Alexandre Lung-Yut-Fon, Frank Portman, Xiao Zhu, Yuanpu Xie, Akshay Gupta, Michael Bronstein, Amra Delić, Gabriele Sottocornola, Walter Anelli, Nazareno Andrade, Jessie Smith, and Wenzhe Shi. *Privacy-Preserving Recommender Systems Challenge on Twitter's Home Timeline*. 2020. arXiv: 2004.13715 [cs.SI] (see pages 31, 33, 49).

[CHM17] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. **Temporal Patterns in Bot Activities**. In: *Proceedings of the 26th International Conference on World Wide Web Companion*. WWW '17 Companion. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, 1601–1606. ISBN: 978-1-4503-4914-7. DOI: 10.1145/3041021.3051114. URL: https://doi.org/10.1145/3041021.3051114 (see pages 53, 55).

[Chu+10] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. **Who is Tweeting on Twitter: Human, Bot, or Cyborg?** In: Dec. 2010, 21–30. DOI: 10.1145/1920261.1920265 (see page 51).

[Cre+16] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi. **DNA-Inspired Online Behavioral Modeling and Its Application to Spambot Detection**. *IEEE Intelligent Systems* 31:5 (Sept. 2016), 58–64. ISSN: 1541-1672. DOI: 10.1109/MIS.2016.29 (see page 54).

[Cre+17] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. **The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race**. In: *Proceedings of the 26th International Conference on World Wide Web Companion*. WWW '17 Companion. Perth, Australia, 2017, 963–972. ISBN: 978-1-4503-4914-7. DOI: 10.1145/3041021.3055135. URL: https://doi.org/10.1145/3041021.3055135 (see pages 53, 56).

[Dav+16] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. **BotOrNot: A System to Evaluate Social Bots**. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. WWW '16 Companion. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016, 273–274. ISBN: 978-1-4503-4144-8. DOI: 10.1145/2872518.2889302. URL: https://doi.org/10.1145/2872518.2889302 (see page 54).

[Dev+18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. cite arxiv:1810.04805Comment: 13 pages. 2018. URL: http://arxiv.org/abs/1810.04805 (see page 44).

[DJ10] Easley David and Kleinberg Jon. **Networks, Crowds, and Markets: Reasoning About a Highly Connected World**. USA: Cambridge University Press, 2010. ISBN: 0521195330 (see page 35).

[Dwi+20] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. *Benchmarking Graph Neural Networks*. 2020. arXiv: 2003.00982 [cs.LG] (see page 39).

[Fer+16] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. **The Rise of Social Bots**. *Communications of the ACM* 59:7 (June 2016), 96–104. ISSN: 0001-0782. DOI: 10.1145/2818717. URL: http://doi.acm.org/10.1145/2818717 (see pages 53, 57).

[Fri02] Jerome H. Friedman. **Stochastic gradient boosting**. *Computational Statistics & Data Analysis* 38:4 (2002), 367–378. ISSN: 0167-9473. DOI: https://doi.org/10.1016/S0167-9473(01)00065-2 (see page 38).

[Ham] William L. Hamilton. **Graph Representation Learning**. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14:3 (), 1–159 (see page 7).

[Hu+20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. *Open Graph Benchmark: Datasets for Machine Learning on Graphs*. 2020. arXiv: 2005.00687 [cs.LG] (see page 39).

[HYL17a] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, 1024–1034. URL: http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf (see pages 56, 57).

[HYL17b] William L. Hamilton, Rex Ying, and Jure Leskovec. **Inductive Representation Learning on Large Graphs**. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 1025–1035. ISBN: 9781510860964 (see pages 8, 45, 46).

[HYL17c] William L. Hamilton, Rex Ying, and Jure Leskovec. **Representation Learning on Graphs: Methods and Applications**. *CoRR* abs/1709.05584 (2017). arXiv: 1709.05584. URL: http://arxiv.org/abs/1709.05584 (see page 57).

[Jia+14] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. **CatchSync: Catching Synchronized Behavior in Large Directed Graphs**. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, New York, USA: Association for Computing Machinery, 2014, 941–950. ISBN: 9781450329569. DOI: 10.1145/2623330.2623632. URL: https://doi.org/10.1145/2623330.2623632 (see page 34).

[KB15] Diederik P. Kingma and Jimmy Ba. **Adam: A Method for Stochastic Optimization**. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980 (see page 47).

[KW16] Thomas N. Kipf and Max Welling. **Semi-Supervised Classification with Graph Convolutional Networks**. *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907 (see pages 45, 55).

[Kwa+10] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. **What is Twitter, a Social Network or a News Media?** In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: Association for Computing Machinery, 2010, 591–600. ISBN: 9781605587998. DOI: 10.1145/1772690.1772751. URL: https://doi.org/10.1145/1772690.1772751 (see page 14).

[LEC11] Kyumin Lee, Brian David Eoff, and James Caverlee. **Seven months with the devils: a long-term study of content polluters on Twitter**. In: *In AAAI Int'l Conference on Weblogs and Social Media (ICWSM)*. 2011 (see pages 54, 56).

[Ma+21] Tengteng Ma, Yuheng Hu, Yingda Lu, and Siddhartha Bhattacharyya. **Graph Neural Network for Customer Engagement Prediction on Social Media Platforms**. In: Jan. 2021. DOI: 10.24251/HICSS.2021.505 (see page 49).

[Mor+21] Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen Carley. **Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose**. *Proceedings of the International AAAI Conference on Web and Social Media* 7:1 (Aug. 2021), 400–408. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/14401 (see page 42).

[Nas+18] Mehwish Nasim, Andrew Nguyen, Nick Lothian, Robert Cope, and Lewis Mitchell. **Real-time Detection of Content Polluters in Partially Observable Twitter Networks**. In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, 1331–1339. ISBN: 978-1-4503-5640-4. DOI: 10.1145/3184558.3191574. URL: https://doi.org/10.1145/3184558.3191574 (see pages 55, 57).

[Pea14] Judea Pearl. **Probabilistic reasoning in intelligent systems: networks of plausible inference**. Elsevier, 2014 (see page 61).

[PMM18] Jürgen Pfeffer, Katja Mayer, and Fred Morstatter. **Tampering with Twitter's Sample API**. *EPJ Data Science* 7:1 (Dec. 2018), 50. ISSN: 2193-1127. DOI: 10.1140/epjds/s13688-018-0178-0. URL: https://doi.org/10.1140/epjds/s13688-018-0178-0 (see pages 15–17, 29).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. **GloVe: Global Vectors for Word Representation**. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162 (see page 44).

[Sch+20] Benedikt Schifferer, Gilberto Titericz, Chris Deotte, Christof Henkel, Kazuki Onodera, Jiwei Liu, Bojan Tunguz, Even Oldridge, Gabriel De Souza Pereira Moreira, and Ahmet Erdem. **GPU Accelerated Feature Engineering and Training for Recommender Systems**. In: *Proceedings of the Recommender Systems Challenge 2020*. RecSysChallenge '20. Virtual Event, Brazil: Association for Computing Machinery, 2020, 16–23. ISBN: 9781450388351. DOI: 10.1145/3415959.3415996. URL: https://doi.org/10.1145/3415959.3415996 (see page 49).

[Sta17] Mark Stamp. **8 Data Analysis**, 228–232. In: Chapman & Hall/CRC, 2017 (see page 38).

[TAM19] Raad Bin Tareaf, Seyed Ali Alhosseini, and Christoph Meinel. **Facial-Based Personality Prediction Models for Estimating Individuals Private Traits**. In: *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, ISPA/BDCloud/SocialCom/SustainCom 2019, Xiamen, China, December 16-18, 2019*. IEEE, 2019, 1586–1594. DOI: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00233. URL: https://doi.org/10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00233 (see page 5).

[TAM20] Raad Bin Tareaf, Seyed Ali Alhosseini, and Christoph Meinel. **Does Personality Evolve? A Ten-Years Longitudinal Study from Social Media Platforms**. In: *IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, ISPA/BDCloud/SocialCom/SustainCom 2020, Exeter, United Kingdom, December 17-19, 2020*. Ed. by Jia Hu, Geyong Min, Nektarios Georgalas, Zhiwei Zhao, Fei Hao, and Wang Miao. IEEE, 2020, 1205–1213. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00179. URL: https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00179 (see page 5).

[Tar+19] Raad Bin Tareaf, Seyed Ali Alhosseini, Philipp Berger, Patrick Hennig, and Christoph Meinel. **Towards Automatic Personality Prediction Using Facebook Likes Metadata**. In: *14th IEEE International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2019, Dalian, China, November 14-16, 2019*. Ed. by Li Zou, Lingling Fang, Bo Fu, and Panpan Niu. IEEE, 2019, 714–719. DOI: 10.1109/ISKE47853.2019.9170375. URL: https://doi.org/10.1109/ISKE47853.2019.9170375 (see page 5).

[Var+17] Onur Varol, Emilio Ferrara, Clayton A Davis, Filippo Menczer, and Alessandro Flammini. **Online human-bot interactions: Detection, estimation, and characterization**. In: *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017*. AAAI Press, 2017 (see pages 53, 56).

[VRA18a] Soroush Vosoughi, Deb Roy, and Sinan Aral. **The spread of true and false news online**. *Science* 359:6380 (2018), 1146–1151. ISSN: 0036-8075. DOI: 10.1126/science.aap9559. URL: https://science.sciencemag.org/content/359/6380/1146 (see page 39).

[VRA18b] Soroush Vosoughi, Deb Roy, and Sinan Aral. **The spread of true and false news online**. *Science* 359:6380 (2018), 1146–1151. ISSN: 0036-8075. DOI: 10.1126/science.aap9559. eprint: http://science.sciencemag.org/content/359/6380/1146.full.pdf. URL: http://science.sciencemag.org/content/359/6380/1146 (see page 51).

[Yan+12] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. **Analyzing Spammers' Social Networks for Fun and Profit: A Case Study of Cyber Criminal Ecosystem on Twitter**. In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12. Lyon, France: ACM, 2012, 71–80. ISBN: 978-1-4503-1229-5.

DOI: 10.1145/2187836.2187847. URL: http://doi.acm.org/10.1145/2187836.2187847 (see page 54).

[YHG13] Chao Yang, Robert Harkreader, and Guofei Gu. **Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers**. *IEEE Transactions on Information Forensics and Security* (2013) (see pages 53, 54, 56).

List of Publications

Articles in Refereed Conference Proceedings

[0] **Engaging with Tweets: The Missing Dataset On Social Media**. In: *Proceedings of the Recommender Systems Challenge 2020*. RecSysChallenge '20. Virtual Event, Brazil: Association for Computing Machinery, 2020, 34–37. ISBN: 9781450388351. DOI: 10.1145/3415959.3415999. URL: https://doi.org/10.1145/3415959.3415999. Joint work with Seyed Ali Alhosseini, Raad Bin Tareaf, and Christoph Meinel.

[0] **Detect Me If You Can: Spam Bot Detection Using Inductive Representation Learning**. In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. San Francisco, USA: Association for Computing Machinery, 2019, 148–153. ISBN: 9781450366755. DOI: 10.1145/3308560.3316504. URL: https://doi.org/10.1145/3308560.3316504. Joint work with Seyed Ali Alhosseini, Raad Bin Tareaf, Pejman Najafi, and Christoph Meinel.

[0] **Predicting User Engagements Using Graph Neural Networks on Online Social Networks**. In: *IThe 8th IEEE International Conference on Data Science and Systems, DSS 2022, Chengdu, China, December 18-21, 2022*. IEEE, 2022, 200–205. Joint work with Seyed Ali Alhosseini and Christoph Meinel.

[0] **The More The Merrier: Reconstruction of Twitter Firehose**. In: *International Conference on Information Networking, ICOIN 2023, Bangkok, Thailand, January 11-14, 2023*. IEEE, 2023, 200–205. DOI: 10.1109/ICOIN56518.2023.10048898. URL: https://doi.org/10.1109/ICOIN56518.2023.10048898. Joint work with Seyed Ali Alhosseini and Christoph Meinel.

[0] **Facial-Based Personality Prediction Models for Estimating Individuals Private Traits**. In: *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, ISPA/BDCloud/SocialCom/-SustainCom 2019, Xiamen, China, December 16-18, 2019*. IEEE, 2019, 1586–1594. DOI: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00233. URL: https://doi.org/10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00233. Joint work with Raad Bin Tareaf, Seyed Ali Alhosseini, and Christoph Meinel.

[0] **Does Personality Evolve? A Ten-Years Longitudinal Study from Social Media Platforms**. In: *IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, ISPA/BDCloud/-SocialCom/SustainCom 2020, Exeter, United Kingdom, December 17-19, 2020*. Ed. by Jia Hu, Geyong Min, Nektarios Georgalas, Zhiwei Zhao, Fei Hao, and Wang Miao. IEEE, 2020, 1205–1213. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00179. URL: https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00179. Joint work with Raad Bin Tareaf, Seyed Ali Alhosseini, and Christoph Meinel.

[0] **Towards Automatic Personality Prediction Using Facebook Likes Metadata**. In: *14th IEEE International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2019, Dalian, China, November 14-16, 2019*. Ed. by Li Zou, Lingling Fang, Bo Fu, and Panpan Niu. IEEE, 2019, 714–719. DOI: 10.1109/ISKE47853.2019.9170375. URL: https://doi.org/10.1109/ISKE47853.2019.9170375. Joint work with Raad Bin Tareaf, Seyed Ali Alhosseini, Philipp Berger, Patrick Hennig, and Christoph Meinel.