A Study of Social Media Trolls via Graph Representation Learning

Albert M. Orozco Camacho

School of Computer Science

McGill University, Montréal

February, 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Science in Computer Science

Abstract

In modern times, social media platforms provide accessible channels for interacting and sharing information about important real-time events. However, these platforms are also regularly targeted by coordinated information attacks. Twitter has publicly released datasets of confirmed fake accounts under their **Information Operations** program to allow researchers to study these attacks. These accounts are commonly known as *Internet trolls*.

In this thesis, we study three of the troll datasets from the *Twitter Information Operations* program, whose origin has been traced to Russia, China, and the Internet Research Agency (IRA). We first augment each dataset with a carefully sampled control group of active users engaged with similar content but not suspended by Twitter. This provides us with a rich set of online posts to study how state-backed trolls behave, how the troll activity fluctuates over time, and how these fluctuations compare to active users. In particular, we use *graph representation learning* to encode users' activities in each timestamp into a link prediction task. These learned representations are then used to contrast troll and active users. We show that, on average, the model has a more challenging time predicting the activity of trolls in two of our three datasets. The model also struggles to classify trolls against active users in these two datasets. However, in the third dataset, trolls are more predictable and easily distinguishable from active users.

We hypothesize that troll sophistication might be related to whether they target local or global events. Finally, we discuss how these representations could help us better understand the activities and how they engage with active users. To show this, we group

link embeddings into clusters, and within each cluster, we contrast the content generated by both categories of users. Although an automatic classification might not be possible for sophisticated trolls, learning user graph representations is, at least, helpful to understand their patterns of activities better.

Abrégé

En temps modernes, les plateformes de médias sociaux pourvoient des canaux accessibles pour l'interaction et réaction aux informations sur des événements importants en temps réel. Cependant, ces plateformes sont également régulièrement la cible d'attaques coordonnées d'informations. Pour permettre aux chercheurs d'étudier ces attaques, Twitter a publié des données de comptes factices confirmés dans le cadre de leur programme d'**Opérations d'Information**. Ces comptes sont communément appelés *trolls Internet*.

Dans cette thèse, nous étudions trois ensembles de données de trolls issus du programme d'Opérations d'Information de Twitter, dont l'origine a été retracée en Russie, en Chine et à l'Internet Research Agency (IRA). Nous augmentons, d'abord, chaque ensemble de données avec un groupe témoin soigneusement échantillonné d'utilisateurs actifs impliqués dans des contenus similaires, mais non suspendus par Twitter. Cela nous fournit un riche groupe de publications en ligne pour étudier comment se comportent les trolls financés par l'État, comment l'activité des trolls fluctue dans le temps et comment ces fluctuations se comparent à celles des utilisateurs actifs. Particulièrement, nous utilisons *l'apprentissage de représentations des graphes* pour encoder les activités des utilisateurs à chaque instant, en une tâche de prédiction des liens. Ces représentations apprises sont ensuite utilisées pour contraster les trolls et les utilisateurs actifs. Nous montrons que, en moyenne, le modèle a plus de difficulté à prédire l'activité des trolls dans deux de nos trois ensembles de données. Dans ces deux ensembles, le modèle a également du mal à classer les trolls par rapport aux utilisateurs actifs. Néanmois, dans le troisième ensemble

de données, les trolls sont plus prévisibles et facilement distinguables des utilisateurs actifs.

Nous émettons l'hypothèse que la sophistication des trolls pourrait être liée à leur cible d'événements locaux ou globaux. Enfin, nous discutons de la façon dont ces représentations pourraient nous aider à mieux comprendre les activités et comment elles interagissent avec les utilisateurs actifs. Pour montrer cela, nous regroupons des vecteurs de liens en clusters de discussion et, dans chaque cluster, nous contrastons le contenu généré par les deux catégories d'utilisateurs. Nous concluons que, bien que la classification automatique ne soit pas possible pour les trolls sophistiqués, l'apprentissage de représentations des graphes des utilisateurs est au moins utile pour mieux comprendre leurs patrons d'activités.

Acknowledgements

To my dear mother. Family bonds, in times when the world has shown a very abnormal face, provide a mental and emotional everyday challenge. Yet my professional growth cannot be explained without your patience and positive attitude. In the middle of a global pandemic, where isolation was the most important asset to keep one's health intact, life allowed us to share valuable moments as I completed my degree back home. Rest assured that my daily motivations came from you and our dog's well-being.

To my dad. A person whom I have learned to appreciate and enjoy some of the finest moments in life. Despite the distance, I love how our relationship has developed on the virtues of empathy, joy, and good sarcasm. If my list of concerts to attend and records to listen to grows yearly, it is mainly because of you. And, of course, I need to recall how life gave us, during the current times, that special moment we have been waiting for a long time.

To that friend who lives in Southern Mexico City, nearby the biggest stadium in the country. (Yes, I did remember to include you in my acknowledgments this time!). After years of conversations that take hours, I still do not understand how our chemistry gave rise to some of the best moments in my life. Speaking of which, you have always been there to witness those, as much as you have been during my downs. As the human being who knows me the best, I appreciate how you have allowed me to become close to some of the most important people that surround you.

To the university professor who taught me AI without knowing how to code. As I always say, you are my academic mentor and the person that most believes in me professionally. To my aunt and my grandparents, whose most valuable asset is life by itself, after many months of mandatory quarantine.

¡Muchas Gracias!

To the sketchy worm guy. My time at Montréal could not be better without your company and empathy. I am grateful for the chance to share our passions while we perform our daily academic endeavors. Yet we know the philosophy of friendly gossiping ought to be maintained every Sunday, at brunch time, for scientific advancements. Thank you for letting me be your roommate for a long time.

To the person who started a fancy vinyl collection and introduced me to the best bar in Montréal. Along with your girlfriend, you were one of the few generous people to share your time with me during those uncertain first months of the pandemic. I will always remember those fantastic nights and parties during my first years in Canada.

To all other people who came close to my life and with whom I developed tight connections. You guys are why we, as graduate students, keep working at the end. To all fellow Mexicans with whom I have formed a community outside our homeland, and to all those who come from other parts of the world and have shared their cultural aspects to enrich our lives.

Merci Beaucoup!

To my supervisor. The person who allowed me to study abroad and who allowed me to discover a personally new way of doing academics. I will always be thankful for the opportunity to open the doors of Mila and, thus, let me discover the fascinating world of machine learning. The road has been difficult, but I am sure I have become a better professional after working with your group.

To all those labmates who always contributed to making graduate school a fun learning process. To all who were there to make me believe in my research and who involuntarily helped me feel more secure and confident about my skills. To those who gave some time and feedback to improve this thesis, and most importantly, to those who were there as emotional support.

Thank You Very Much!

Contents

Abstract i
Abrégé iii
Acknowledgements v
List of Figures xi
List of Tables xii

1 Introduction **1**

2 Related Work **7**
2.1 Graph Theory 7
2.1.1 Graphs, Digraphs, Matrices 8
2.1.2 Paths and Distances 14
2.1.3 Graphs and Their Components 16
2.2 Network Science 18
2.2.1 Properties of Large Networks 19
2.2.2 Homophily and Affiliation 21
2.3 Machine Learning for Graphs 22
2.3.1 Common ML Tasks for Graphs 24
2.3.2 Engineering Graph Features into Vectors 26
2.3.3 Node Embeddings 27
2.3.4 Graph Neural Networks 29
2.4 Link Prediction 34

2.4.1 **S**ubgraphs, **E**mbeddings, and **A**ttributes for **L**ink **P**rediction 37
2.5 Misinformation Spread 39
2.5.1 Fake News 39
2.6 Troll/Bot Detection 41
2.6.1 Understanding Verified Trolls 42
2.7 Sampling Social Media Datasets 43
2.7.1 Sampling Data from Twitter 44

3 Methodology **46**
3.1 Modelling Troll Activity with Graphs 46
3.1.1 The Twitter Information Operations Dataset 46
3.2 Activity Collection Process and Data Augmentation 47
3.2.1 Sampling Active Users 48
3.2.2 Constructing Mention-Hashtag Graphs 50
3.3 Analyzing Dynamic Mention-Hashtag Graphs 55
3.3.1 Representation Learning of User Activity 55
3.3.2 Activity Prediction and User Classification 58
3.3.3 Clustering User Activity 60

4 Experiments **62**
4.1 Experimental Data Preparation 62
4.2 Predicting Behaviour through Link Prediction 67
4.3 Classifying Users through Node Classification 72
4.4 Experimental Interpretations 75

5 Conclusion **86**
5.1 Summary of Work 86
5.2 Contributions 89
5.3 Future Work 91

5.4 Concluding Remarks 92

A Further Dataset Descriptions **107**

A.1 Monthly Statistics 107

A.2 All Time Troll Activity 107

B Summary of Experiments **111**

B.1 Experiment Settings 111

B.2 More on Link Prediction 111

B.3 More on Node Classification 114

List of Figures

1.1 Sample TEI troll activity 4

2.1 The Bridges of Königsberg 8
2.2 Types of Graphs 9
2.3 Adjacency Matrices 13

3.1 Network visualization of mention-hashtag graphs 50
3.2 Network evolution over time 51
3.3 Mention-hashtag graph 52
3.4 Illustration of the chosen metapaths 56
3.5 Tweet-to-embedding preprocessing pipeline 57

4.1 Active user sample process 64
4.2 Activity per time 66
4.3 Graph Sender vs Receiver CDF 68
4.4 Proportion of correctly predicted links per each place of origin 71
4.5 Link prediction score fluctuation per time 73
4.6 Comparison of Link Prediction and Node Classification scores per time . . . 78
4.7 Discussion Topics as Link Embedding Clusters 82

A.1 Monthly Tweet Activity, Russia 108
A.2 Monthly Tweet Activity, IRA 108
A.3 Monthly Tweet Activity, China 109

A.4 All Time Troll Activity 110

B.1 Experiment Settings 112
B.2 LP Accuracy per active user dataset, Russia 113
B.3 LP Accuracy per active user dataset, IRA 114
B.4 LP Accuracy per active user dataset, China 115

List of Tables

2.1 Different Aggregation Methods 31

3.1 Statistics of datasets 49
3.2 Clustering Variables 61

4.1 Receiver node percentage out of total available 67
4.2 Link prediction performance scores 69
4.3 Link prediction scores, without node labeling 70
4.4 Node classification performance scores 74
4.5 Average Link Prediction Pearson Correlations 77
4.6 TEI Data Important Events 81
4.7 Word Clouds Russia 83
4.8 Word Clouds IRA 84
4.9 Word Clouds China 85

B.1 Node classification Accuracies 116
B.2 Node Classification F1 Scores, Node Labeling Ablation 116

Chapter 1

Introduction

We live in times where social media platforms have risen as pillars of the spread of information. Their success can be accounted for as they provide a space for individuals and organizations to share ideas, news, and updates with a large audience quickly and easily. However, oftentimes, certain users take advantage of the rapid and efficient information sharing on these platforms; thus, making the proliferation of massive false narratives possible. For instance, a plethora of social media campaigns have sprung up on **Twitter**[1], a popular short-posting platform, to deny reports of human rights abuses committed by the Chinese government against Uyghurs and other minority groups in Xinjiang, often involving spam networks of similar-looking accounts posting identical messages (Kao et al., 2021). This is especially problematic since Twitter has one of the largest user bases on the Internet (Dixon, 2022). But how could we study large-scale phenomena of shifting public opinion in terms of small-scale events, such as social media posts? Is that even possible?

Before going through the latter question's importance, let us take a look at one of the 21st century's global concerns: *the COVID-19 pandemic*. As people's daily lives suffered a sudden transition due to mandatory isolation policies, guaranteeing the flow of reliable information turned crucial as social networks became one of the main ways to

[1] `https://twitter.com`

keep updated about the pandemic's development. Unfortunately, this derived into a phenomenon, where the spread of misinformation via conspiracy theories and other unsubstantiated rumors formed an online infodemic (Evanega et al., 2020; Gallotti et al., 2020; Memon and Carley, 2020; Sharma et al., 2020). Here we land into an unsolved, and sometimes even ill-defined issue: how does one identify online misinformation, such as fake news? Even harder, and much less explored: *who is behind this unreliable activity and what are the main strategies they deploy to attract the attention of users?* Perhaps by better understanding the mechanism of misinformation, it would be possible to design better algorithms to track it (Jin et al., 2014; Memon and Carley, 2020; Pennycook et al., 2020; Wang et al., 2019). We align with this hypothesis and work towards it in this thesis.

In principle, Twitter tries to provide a platform for free speech[2]; nonetheless, that has also resulted useful for the appearance of user infringements. It can be argued that the current times require tighter and clearer online interaction rules, which would address misinformation spread. In fact, some practices as old as spamming are supposed to be regulated for the sake of ensuring better authenticity of statements and their actual motives. More sophisticated unauthentic activities have been deployed to manipulate public opinion, which often constitutes a crucial practice by politically-driven organizations. For example, the *Russian troll farm* from **Internet Research Agency** (IRA) has been targeting the United States presidential elections in 2016, as confirmed by multiple reports (Volchek, 2021; Wong, 2020). Moreover, the IRA's communication strategy has evolved to account for the developments in misinformation tracking deployed by Twitter (Alba, 2020).

In this thesis, we focus on studying *troll behavior* and understanding the mechanisms employed by them. The term **troll** now has a history of standard usage within the Internet. There are two closely related concepts to what we refer to as trolls: *bots* and *cyborgs* (Klepper, 2020). On one side, *bots* are considered to be fully automated accounts that post predefined content, *often using spamming practices to* which deliver an abnormal amount of tweets, compared to humans. In contrast, *trolls* are accounts entirely run by humans,

[2] `https://help.twitter.com/en/rules-and-policies/defending-and-respecting-our-users-voice`

which also follow posting practices and align with "normal" patterns of created information.

Cyborgs, on the other hand, constitute a combination of the earlier approaches, as humans occasionally take over bot spammers to reply and post content themselves. For this thesis, we focus on those accounts that are harder to identify than fully automated bots, whose decision strategies are determined by *coordinated attacks*. These attack operations often occur within the context of professional groups that often post controversial content to social networks, called *troll farms* (Hao, 2021).

In particular, we consider accounts released by the **Information Operations** program of Twitter as trolls. As a means to encourage research, Twitter has periodically released all *tweet* post content (including attached media) of, supposedly, several state-backed operations. Starting on October 2018, these databases have provided ubiquitous insights to understand the type of content that often derive from misinformation or massive coordinated attacks on political events. As part of their **Information Operations** strategy, the company's transparency efforts have also made these data batches accessible for free, with easy access directly from their website (Inf, 2021). The **Twitter Election Integrity** (*TEI*) is, hence, an *official* collection of account activity that would make evident how *trolls* are operating online. Each release has been identified with its respective *place of origin* and comes together with a brief explanation of what led to the suspension of these accounts. For instance, the first release (Gadde and Roth, 2018a) included data from two different sets of accounts: $3,841$ IRA-affiliated suspended and 770 Iranian users, both producing a combined amount of over *10 million tweets* and *2 million* images and videos. Figure 1.1 provides graphical examples of some of the content posted by these trolls.

Combining this vast amount of information to study massively coordinated accounts is the pinnacle of this thesis' research purposes. As mentioned earlier, we take a modern approach to analyze parts of the data within the context of state-of-the-art Deep Learning (DL) models. More specifically, we are concerned with two central questions: *are these troll behaviors more predictable than the ordinary human online traces?* and *can we learn a model*

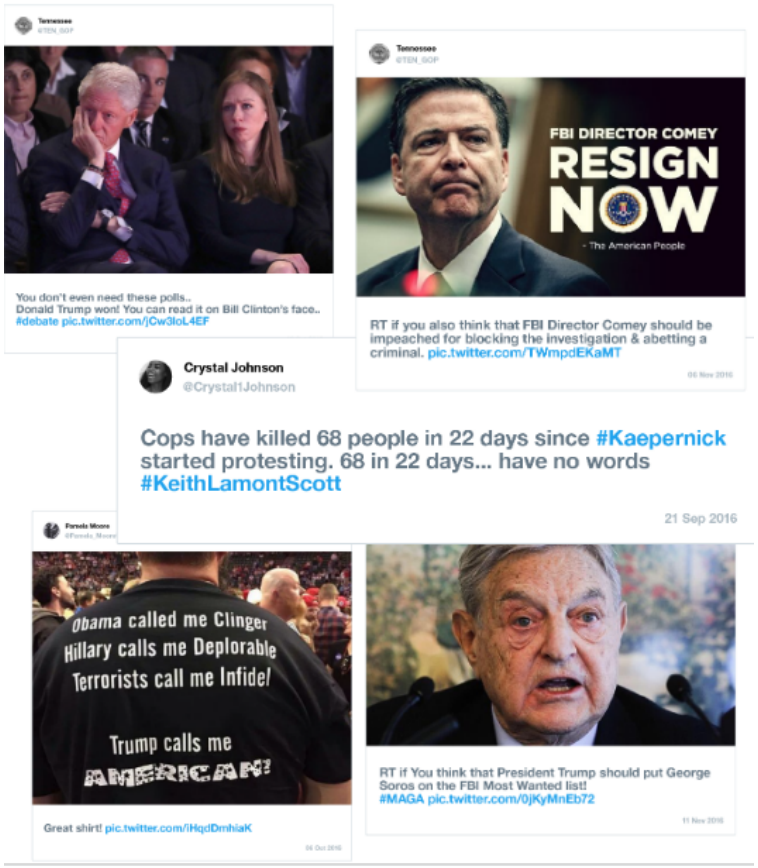


Figure 1.1 – Sample tweets posted by accounts from the Twitter Election Integrity dataset. Most of this content is designed to inject controversial opinions that would provoke polarizing reactions among Twitter users. Taken from `https://about.twitter.com/en_us/values/elections-integrity.html#data`.

that classifies these trolls solely from features that explain their behavior? To further formalize these questions, we take a **Graph Representation Learning** (GRL) approach and provide a collection and processing pipeline to reconstruct some of the relationships established by these users at times they were active. More specifically, in this thesis:

- We utilize three different data releases from the TEI dataset, namely one with *Chinese* users, another one with *IRA* users, and finally, a *Russian*-based set of users (but not originating at the IRA), as examples of troll activity (these users have already been **suspended**); we complement These troll datasets with a method to obtain a set of *non-suspended* users, which serves as a control group. We provide two neighborhood

sampling methods for active users to obtain more distant accounts (with respect to user mentions) while making sure that each sampled tweet relates somehow to the trolls.

- We model our data as a discrete-time temporal graph. In particular, we construct heterogeneous user-mention-hashtag graphs to represent user activity within each snapshot. We study troll activities over two years about, mainly, political events. We consider four ranges for the time granularity to take the snapshots (five, ten, thirty, and sixty days) and show how this granularity impacts the analyses.

- We encode user activities in each snapshot independently, using graph representation learning techniques. In particular, we train the models using link prediction loss functions, which learn the distribution of certain activity (user mentions and hashtag use), commonly formulated and used for the link prediction (LP) task. This model keeps closer users that act similarly in each snapshot.

- Using the learned user representations in each snapshot, we track how accurately a user's activities can be predicted throughout time and if trolls are more or less predictable than the control group. We also use these representations to see if we can directly classify and detect suspended trolls. We show that the automatic classification task is difficult in two of our three datasets with Russian sources whose activities are also harder to predict.

- Finally, we zoom in on critical snapshots tied to major electoral events in each dataset. Therefore, we show how these learned embeddings could help understand the engagement of trolls and the control group. Moreover, we have been able to interpret these clusters of activity embeddings by looking at topics users are talking about; in particular, our embeddings can group semantically related topics, even though our approach follows a procedure agnostic to text features.

We further organize this thesis to provide the necessary background and support for our work adequately. Chapter 2 builds up from the fundamentals of *graph theory* to current ways these abstract items are processed by DL models while also mentioning their relationship to our proposed work. Chapter 3 introduces our methodology, which is defined as a processing pipeline that connects extracted information to classification outcomes. Furthermore, in Chapter 4, we put into practice a pipeline with a set of experiments that would help us understand how it works and answer our hypotheses. Finally, we round up our work on Chapter 5.

Chapter 2

Related Work

2.1 Graph Theory

We start this chapter with a brief introduction to graph theory. Moreover, we provide the background needed to justify this thesis. To gather the necessary mathematical formality together, for the sake of this thesis, we have adapted this section's content from (Barabási and Pósfai, 2016, Chapters 2, 3) and from (Easley and Kleinberg, 2010, Chapters 2, 4).

While interconnected phenomena have existed forever, the formality of graph theory stems its motivation from the formulation and solution of the famous **seven bridges of Königsberg** problem. In 1735, this Prussian city's[1] trading activity required officials to build *seven bridges* across the River Pegel to connect the mainland with Kneiphof island. The curious question posed was: *could it be possible to walk across all seven bridges without crossing the same one twice?*

Later, the illustrious mathematician, Leonhard Euler, came up with proof that guarantees no solution for such a problem. Despite several clever attempts to develop a path between bridges and land ends, Euler's graph formalization guaranteed its non-existence. Euler's technique transformed Königsberg's mainlands into nodes while defining connections corresponding to the number of edges in between. Figure 2.1 summarizes Euler's

[1] Königsberg constitutes what currently is known as Kaliningrad, Russia

idea visually. Graphs exist naturally within diverse settings; examples like *Euler's Königsberg proof* provide remarkable evidence of the power of formalizing and studying graph properties, such as the number of shared links between nodes, the existence of paths between nodes, or the areas of a network where a node is more closely connected.

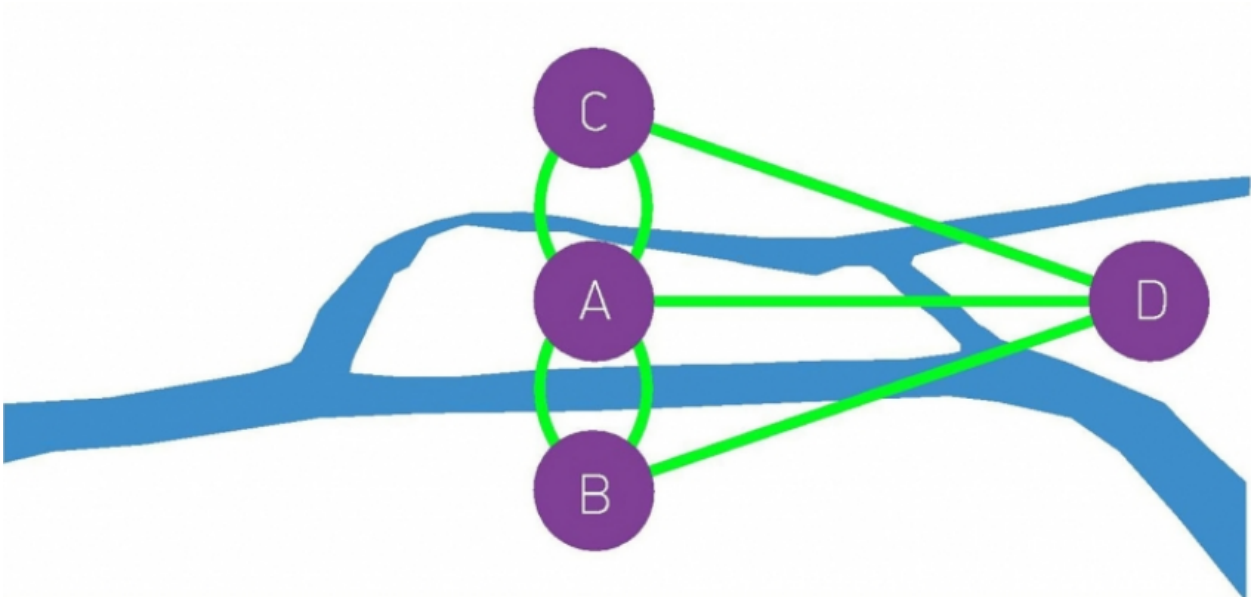


Figure 2.1 – A map of Königsberg and its Kneiphof island is depicted along with the graph proposed by Euler. Here, nodes are defined by each patch of land and are connected with seven links (bridges). Reprinted from Barabási and Pósfai (2016).

2.1.1 Graphs, Digraphs, Matrices

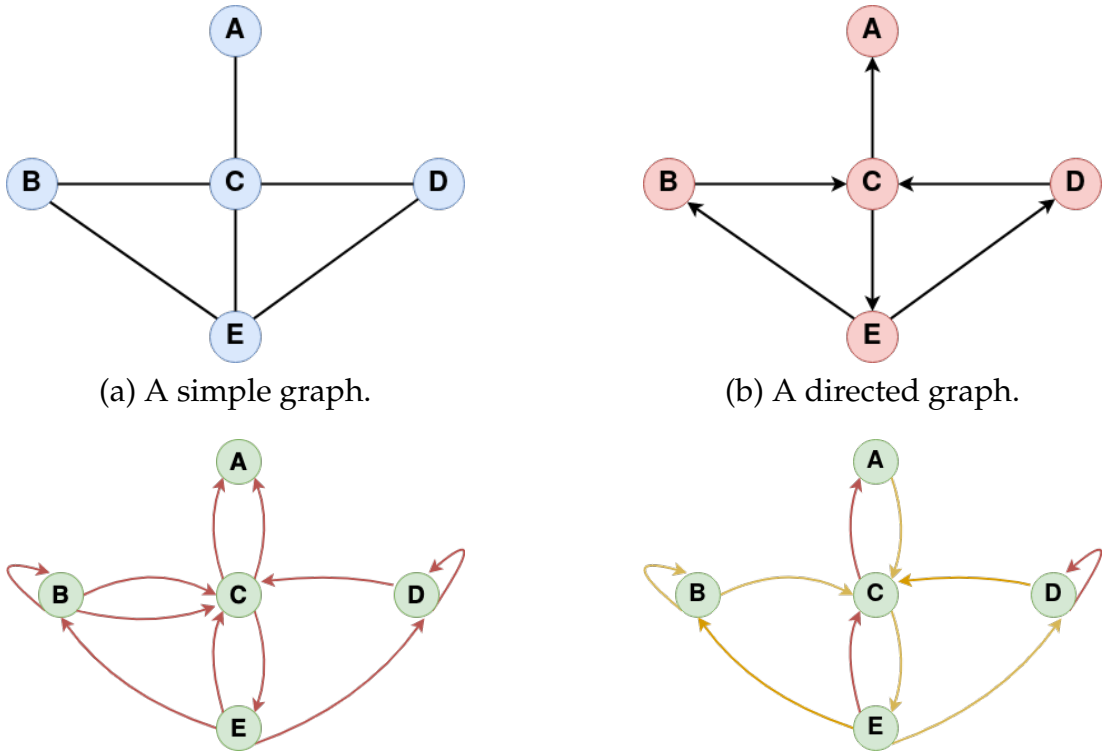
In this section, we formalize the notion of graphs and digraphs and provide helpful notation and essential properties about the subject.

Definition 2.1.1 (Simple Graph)**.** *A simple (or undirected)* ***graph*** G *is defined by a (finite) set* V *of vertices (or nodes) and a set* $E \subseteq \{\{u, v\} \mid (u, v) \in V \times V\}$*. We, thus, write* $G = (V, E)$*.*

Note that every edge, implies a *symmetric relationship between nodes*, as they satisfy the usual set property $(\{u, v\} = \{v, u\})$. Using this abstraction, we may encode for the Königsberg problem, for instance, the fact that Kneiphof island is connected with any surrounding patches of land. However, there is still no way to specify in what order a

person visits the patches while traveling around. This yields the necessity of defining multiple types of graphs.

Definition 2.1.2 (Digraph, Multigraph)**.** *A **digraph** (or directed graph) G consists of a (finite) set V of nodes and a set $E \subseteq V \times V$ of edges (also known as arcs or links), which consist of ordered pairs. A **multigraph** $G = (V, E)$ consists of a set of vertices V and a multiset of pairs of vertices (edges); thus, a multigraph allows multiple edges between two nodes. A **heterogeneous graph** is a multigraph that can encode multiple relations between nodes; that is, each edge $e_l \in E$ has a label l that identifies its relation type.*



(a) A simple graph.

(b) A directed graph.

(c) A multigraph.

(d) A heterogeneous graph.

Figure 2.2 – We show examples of the three main types of graphs presented in this section. Figure (a) is a **simple graph**, while Figure (b) adds arrows to stress the importance of *arcs' directions*, thus, presenting a **directed graph**. Figure (c) is a **multigraph**, where we show that nodes may have more than one arc, such as C to A. Finally, Figure (d) depicts a *heterogeneous graph* in which we allow multiple types of relations (illustrated using different colors) between nodes.

Definitions 2.1.1 and 2.1.2 are illustrated by Figure 2.2, where we show an example for each type of graph. Note that for Figure 2.2-c, the multigraph, we have also used directed edges; note that, indeed, Definition 2.1.2 allows such flexibility. Moreover, colors in the referred example help define the *types of relationships* each pair of nodes share (Figure 2.2-d). This often comes useful in many applications, as we will see throughout this thesis.

The essence of Euler's negative proof of the Königsberg bridges' problem arises by observing the number of edges a node shares with the rest of the graph. In general, we call this property the *degree* of a graph, which is defined in terms of each node's **degree**, as will be formalized in Definition 2.1.3.

Are there any alternative ways of representing a graph or network besides the $G = (V, E)$ formality? Note that a list of all edges suffices to do so. For instance, we may write a code that manages Figure 2.2-a by just specifying the following:

```
[(A, C), (B, C),
 (B, E), (C, D),
 (C, E), (D, C),
 (D, E)].
```

We may augment this list with other pairs to specify digraphs, such as the ones in Figure 2.2. Nevertheless, a universal way to represent any graph arises from observing how to organize and count all the possible connections a node may have.

Definition 2.1.3 (Degrees, Neighborhood, and Average Degree)**.** *Given a simple node u of a graph $G = (V, E)$, its **degree** k_u is defined as the number of edges it has to other nodes:*

$$k_u \equiv |\{(u, v) \in E \mid \forall v \in V\}|$$

- *The set $\{v \in V \mid \exists\, (u, v) \in E\}$ is called u's neighborhood (and the elements, its neighbors).*
- *The degree m of graph is its total number of edges. For simple graphs, this can be obtained by adding up all node degrees:*

  $$m \equiv \frac{1}{2} \sum_{i=1}^{N} k_i,$$

  where N is the total number of nodes. Note that the factor $1/2$ corrects that every link is counted twice.
- *The average degree of a graph, often written as $\langle k \rangle$, is defined as the arithmetic mean over all degrees of the graph:*

  $$\langle k \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} k_i = \frac{2m}{N}$$

Definition 2.1.4. *Given a graph $G = (V, E)$, where $|V| = N$, we define its **adjacency matrix** $A \in \mathbb{R}^{N \times N}$ to be the one that*

$$A_{ij} \equiv \begin{cases} 1, & \text{if } (i, j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

Note that the degree k_i of every node i can be obtained directly from the adjacency matrix. For undirected graphs, we sum over all columns or rows of it:

$$k_i = \sum_{j=1}^{N} A_{ij} = \sum_{j=1}^{N} A_{ji} \tag{2.2}$$

For directed graphs, we distinguish the *incoming* from the *outgoing* degrees by summing over rows or columns, respectively:

$$k_i^{in} \equiv \sum_{j=1}^{N} A_{ij}, \tag{2.3}$$

$$k_i^{out} \equiv \sum_{j=1}^{N} A_{ji} \tag{2.4}$$

In practice, we may also refer to the quantities specified in Equations 2.3 and 2.4 as the *in* and *out* degrees of a digraph G. As for multigraphs, the reader can bear in mind that we can use as many adjacency matrices as the number of relationships a graph is given; hence, we can define degree features per each relation or whole. For instance, Figure 2.2-c has 3 types of relationships. Hence, we may use a $5 \times 5 \times 3$ tensor to encode all adjacencies together.

By distinguishing each in-or-out direction of each arc, we may also specify every digraph's adjacency matrix as two different ones with the assumed convention. Figure 2.3 displays some examples of adjacency matrices corresponding to the examples of Figure 2.2. In particular, we show both incoming and outgoing matrices for Figure 2.2-b. By convention, some literature prefers the *outgoing representation as the default adjacency matrix for any digraph*; hence, its degree features will be determined upon it.

We are now in the position of defining an essential matrix in graph theory: the *Laplacian*. Many of the classic network science contributions have been made possible by exploring the properties of the graph's Laplacian matrix; in particular, its eigenvalues can be interpreted as clustering features that treat any adjacency matrix as a list of *similarity* measures between the nodes.

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

(a) Figure 2.2-a's adjacency matrix.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

(b) Figure 2.2-b's *outgoing* adjacency matrix.

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

(b) Figure 2.2-b's *incoming* adjacency matrix.

Figure 2.3 – Example of the adjacency matrices for some of the graphs in Figure 2.2. Nodes are sorted in alphabetical order from up to down (rows) and left to right (columns).

Definition 2.1.5 (Laplacian Matrix)**.** *Let $G = (V, E)$ be a graph and A its $n \times n$ adjacency matrix. Let D be the diagonal matrix that stores all the degrees of the nodes in G, that is, $D \in \mathbb{R}^{n \times n}$, $D_{ii} = k_i$ for every $i \in V$ while $D_{ij} = 0$ whenever $i \neq j$. The Laplacian matrix L of G is defined as the difference between the diagonal matrix D and A:*

$$L \equiv D - A$$

Let A be the adjacency matrix in Figure 2.3-a. Let D be the diagonal degree matrix for the referred graph that is,

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix} \tag{2.5}$$

Then, the Laplacian matrix results by subtracting A from D:

$$L = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 2 & -1 & 0 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{pmatrix} \tag{2.6}$$

The *connectivity of nodes* results in an essential and valuable relation that drives research in graphs. Within a social network, for instance, a user's main interests are often deduced by looking at their closest friends, which are part of his or her neighborhood. The so-called *structural* features introduce convenient ways to organize and analyze information, with fantastic convenience when network sizes scale while preserving fundamental abstract properties. Such features are produced by diverse kinds of users who interact with each other.

To model interactions between humans and bots, for instance, we might want to distinguish nodes by assigning a *type*. A *bipartite graph* (Definition 2.1.6) provides a fundamental abstraction to achieve this distinction, which is naturally applicable in diverse settings. Note that this definition can be generalized to the concept of k-partite graphs, where we would be able to model the distinction of k types of nodes, which may result convenient when modeling complex data.

Definition 2.1.6 (Bipartite Graph)**.** *Let $G = (V, E)$ be a graph. We call G **bipartite** if V can be partitioned into two disjoint sets A, B, such that $(u, v) \in E$ if and only if $u \in A$ and $v \in B$, for any arbitrary nodes $u, v \in V$.*

2.1.2 Paths and Distances

A bipartite graph can give a group of nodes a common label, depending on who they are connected to. On the other hand, connectivity matters not only on the neighborhood

level but often *further neighbors* are the ones giving more information to a single node[2]. For example, when links encode actual distances between cities in a map, we would like to know the *shortest way* to get from Montréal to Mexico City. Formally, Definition 2.1.7 summarizes some ways of referring to a graph *traversal*.

Definition 2.1.7 (Walk, Trail, Path, Cycle)**.** *Let $G = (V, E)$ be a graph. A **walk** of length l is a sequence of nodes $(v_0, v_2, \ldots, v_{l-1})$ such that $(v_i, v_{i+1}) \in E$ for any $0 \leq i \leq l$.*

*Given a walk $\mathcal{W} = (v_0, v_1, \ldots, v_{l-1})$, we call it **trail** if for any edges $(v_i, v_{i+1}), (v_j, v_{j+1}) \in \mathcal{W}$, if $v_i = v_j$ and $v_{i+1} = v_{j+1}$ then $i = j$ for any $i, j \in [0, l)$, i.e., there are no repeated edges.*

*We call $\mathcal{W}$ a **path** if for any nodes $v_i, v_j \in \mathcal{W}$, if $v_i = v_j$ then $i = j$, i.e., there are no repeated nodes. If $v_0 = v_{l-1}$ we call $\mathcal{W}$ a **cycle**.*

Many problems depend on the existence of paths; in particular, the *traveling salesman problem* (TSP) shows that algorithms will not always be tractable. Given a point A graph, TSP would like to know the shortest path that passes through all nodes and returns to A. It has been famously proven this problem to be *NP-complete*.

As network sizes increase, the tractability of any algorithm raises its importance. It is moreover, not always possible to deal with the whole adjacency matrix of a graph in practice; nevertheless, local connections to a node could allow us to *learn more about it*. In many applications, where we assume to explore a node's neighborhood be the only tractable way to use the graph, we might need a way of exploring a network without fully processing it. This can be achieved through a walk's stochastic generalization that requires a probability distribution to decide which node to go to next.

[2]In practice, we use the term *2-hop neighborhood* of node v to the set of neighbors of v's neighbors all together, that is, $\{u' \| u' \in N(u),\ \forall u \in N(v)\}$. This concept is often generalized to the **k-hop neighborhood** with similar meaning.

Definition 2.1.8 (Random Walk). *Let $G = (V, E)$ be a graph, let π^V be a probability distribution over all nodes V, and P_v be the probability of sampling one of v's neighbors. A **random walk** is a sequence of random variables $X_0, X_1, \ldots, X_{l-1}$, each of them taking a node $x_i \in V$, that result from the following procedure:*

1. *Sample a node from π^V and assign it to X_0.*
2. *Let counter $i \leftarrow 1$*
3. *Sample a node from $N(X_i)$ with probability P_{X_i} and assign it to X_{i+1}*
4. *Increase counter $i \leftarrow i + 1$.*
5. *Repeat steps 3 and 4 until $i = l$.*

Definition 2.1.8 has no prerequisites on which π^V to choose for better use. Bear in mind that, in the fairest scenario, we can take the *uniform* distribution on any set of nodes. In situations where *spurious or unknown* links occur, learning a *random walk* distribution, conditioned on the whole set of nodes, will likely be a great idea to figure out what is a node's *role* within the overall structure of a graph.

2.1.3 Graphs and Their Components

We now go back briefly to a more *deterministic* setting to provide some useful definitions.

Definition 2.1.9 (Subgraph, Induced Graph). *Let $G = (V, E)$ be a graph. A **subgraph** of G is any graph $G' = (V', E')$ where $V' \subseteq V$ and $E' \subseteq E$ requiring that every edge $e \in E'$ has its respective incident nodes in V'. We say that a subgraph G' of G is **induced** if all edges that connect the G''s vertices originally in G are also in G'.*

Formally, we can also talk about graphs that are connected as a whole; while for those that are not, we can also work with their *connected components*. This allows us to think at a higher level where connected subgraphs matter more than individual node connectivity.

Definition 2.1.10 (Connected Graph, Connected Components)**.** *Let $G = (V, E)$ be a graph. We say that G is **connected** if for any nodes $u, v \in V$, there exists a path that connects them. If a graph does not satisfy the previous definition, we distinguish each maximal induced subgraph that is connected by itself; we call these the **connected components** of the graph.*

We know how local connections interact by dividing a graph into its connected components. Sometimes it will not be enough to analyze the connected parts of a large network. While complexity scale retakes a critical role in providing efficient algorithms, we may *relax* the notion of connectivity now to distinguish which parts of a graph are *more connected than others*. This is of particular interest when studying network robustness, as well as getting a glimpse of a person's closest friends in a social setting. Section 2.2.1 will elaborate on the latter situation.

Definition 2.1.11 (Tree, Forest)**.** *A graph is a **tree** if it is connected and has no cycles. If a graph has no cycles, we call it a **forest**.*

Definition 2.1.11 brings in two other important types of graphs and a structure that one can construct when traversing any graph. A *tree* allows us to find a way to organize all the nodes of a graph in such a way that by traversing it, we do not repeat previously scanned structures. For directed graphs, we often call *children* to the out-neighbors of a node v, while its in-neighbors will be its *parents*.

For a node r, called the *root*, there are two important methods to explore any graph by computing its traversal tree. Its fundamental difference relies on the order in which a node's neighbors are explored: while we can proceed to look at all neighbors at each time step, we can also find the furthest node from r and recursively observe its parents. Algorithm 1 (BFS) describes the popular **breadth first search** process in which a graph's nodes are completely explored from a given *root node*, while keeping track of neighborhoods inside a queue.

Algorithm 1: Breadth First Search

Input: Inputs are a graph $G = (V, E)$ and a *root* node $r \in V$

```
q ← EMPTY_QUEUE;
q.enqueue(r);
explored ← EMPTY_SET;
explored.add(r);
while q is not EMPTY_QUEUE do
    v ← q.deque();
    ns ← v.neighbors \ explored;
    explored ← v.neighbors ⋃ explored;
    q.enqueue(ns);
end
```

2.2 Network Science

Most concepts and methods presented in Section 2.1 are meant to characterize the properties of a graph regardless of its use. More concretely, large networks require to be studied on a larger scale; for instance, grouping similar nodes under specific criteria often goes beyond determining the connectivity of a graph. Under this context, finding paths from any arbitrary pair of nodes will not be only a retractable procedure for any *clustering* algorithm. Still, it will dismiss any additional information encoded by the links themselves.

We now focus on quantifying large networks' properties as a whole. Some of them will be regarded as mere *heuristics* that would make sense in most use cases of networks. Other ones will be inspired by interdisciplinary research, such as sociology, and will help us learn from networks. Note that from now on, we will talk more about *networks* and their applications rather than abstract graphs.

Definition 2.2.1 (Clustering Coefficient)**.** *Let $G = (V, E)$ be a graph. For a node v with degree k_v, let L_v be the number of edges that connect each node of the neighbors of v, that is, the degree of the induced subgraph with nodes $N(v)$. The* ***clustering coefficient*** *of v is defined as*

$$C_v \equiv \frac{2L_v}{k_v(k_v - 1)}$$

The *clustering coefficient* gives us a rough idea of how likely it is for a node to live within a *very dense neighborhood*. To give more intuition, consider a dynamic situation where a

new user on Facebook has arrived and is looking to *be-friend* his or her acquaintances. Regardless of the company's successful recommendation strategies, the user might proceed to identify which of his or her contacts are the most popular ones; by *be-friending* them, it is very likely to access other acquaintances from their respective Facebook friendship relationships. We may argue that this user used a *clustering coefficient* comparison to access its acquaintances quickly.

2.2.1 Properties of Large Networks

The discussion of *how networks grow* is large enough to dedicate a whole scientific field to itself. On the other hand, we shall consider this context instead to study *why* these networks form patterns. For starters, sociologists have been looking at why two people are becoming friends, which has led to interesting conclusions. From a global perspective, it is not complete randomness that drives the social network's link size to increase (or to *shrink*). Instead, and even intuitively, people look for closest friends to acquire new friendships with their closest acquaintances. This idea is summarized by the *triadic closure principle*.

Principle 2.2.2 (Triadic Closure)**.** *If two nodes in a social network have a common neighbor, then it is very likely that they will get connected at some point.*

This way of justifying how links exist within a network has been extensively studied in sociology (Harmon, 1959; Newman, 2003; Rapoport, 1953). It makes sense to assume the friends of my friends are *trustworthy* enough to *be-friend* them. Observe that the clustering coefficient of a node merely provides a numerical way to measure the triadic closure of a network. Moreover, it is worth stressing that this principle will only explain the existence of *a significant* number of links, while some might follow other properties.

As friendships establish a *channel for communication*, studying how information propagates across these links is interesting. After two people on social media finish chatting with each other, *how will the exchanged information affect the rest of their network?* In a large social network, a person's position within a local subgraph might determine *a level of*

tightness of his or her links within the whole graph. It is, thus, important to identify which parts of a graph are *further away*, the ones that are *more central*, and the ones that *link different areas* of a network; each area has a particular role within the network's behavior in dynamic settings.

In particular, those edges whose existence directly affects the connectivity of a graph as a whole are called **bridges** since their absence will create more connected components. If we let some information flow across a network's edges, bridges will be the ones that guarantee its spread across all nodes; moreover, they shall give information on how data propagates and its time frame. Ultimately, bridges help us to identify some of the most fragile parts of a graph; for instance, the possible *bottlenecks* for potential damages to an actual network, which is what *percolation theory* looks for while establishing robustness properties.

How do we verify that a given link is a bridge, in general? Strictly by its definition, any algorithm might not scale as it would rely on finding paths between any pair of edges. Hence, we also rely on a weaker notion of graph connectivity: a **local bridge**. For these kinds of edges, it suffices to guarantee that their removal has increased the distance between two nodes (where distance is measured as the number of nodes of the shortest path between two given nodes). For very complex graphs, where the *small world phenomenon* implies relatively short distances between any nodes, a bridge is an important finding to characterize and understand the network as a whole; in fact, the guarantees of such phenomenon also imply the significance of **local bridges**, as they are directly responsible for separating nodes from each other.

In the context of information flow, people in a social network are, typically, not only agents of transmission. Other aspects, such as linguistic manners, affect *how* are messages propagated and often reflect on the *influence* of certain users towards certain parts of the graph. Examples of influential nodes are quickly drawn from user popularity and engagement, which can be quantified in settings like *Twitter* via the number of followers. On the other hand, neighborhood sizes could not always be a sufficient condition for any

opinion, *rumor*, or *meme* to acquire levels of *virality*. More specifically, when the state of belief of a network changes uniformly about a particular opinion, we say people are *following the herd*.

The social consequences of rumor spread and the structural and external properties that produce herding are extensively studied in the literature (Banerjee, 1992; Bikhchandani et al., 1992). In practice, we refer as information **cascades** to the types of graphs that signal how a rumor is spread. Intuitively, herding behavior may emerge on a set of *seed nodes* that have decided to adopt an external way of thinking rather than continue following its beliefs. Observe that cascades are trees since they depict only the nodes that are switching their beliefs, and then they will not change anymore (hence, nodes are not repeated). Moreover, assembling a network to facilitate rumor spread might have negative consequences; for instance, malicious organizations may set up a global online network for disinformation spread that directly polarizes a country's voter's opinions. The key is, thus, the careful manner in which such organizations decide to target, which shall be identified and reinforced.

2.2.2 Homophily and Affiliation

The triadic principle provides a structural intuition of why nodes connect, yet there are many more reasons this happens beyond *having a familiar friend*. Using the triadic principle, does that imply that new friends are similar in all aspects when people meet? We now look into a more general property that works under the context of the network itself, that is, over the set of features that characterize each node, from which we may compare one with the other.

In a setting where **homophily** is assumed to justify the nodes' links, a user will tend to be-friend *only similar people*. This is, in fact, an old idea that has come to justify behavior nowadays, even to the extent that it has impacted modern research in sociology. If we try to visualize a network that exhibits homophily, we will find that many similar

nodes are all together in the same part. This does not imply similar nodes form connected components; instead, it could be a way to *cluster* the data.

When a network exhibits homophily, most nodes incident to a single edge should have similar characteristics. For simplicity, let's assume both nodes are labeled with the same tag from a set of 2 possible tags to classify each node. Thus, the proportion of edges whose incident nodes are tagged the same must dominate the one with *heterogenous* tags, which implies a method for testing homophily, at least under the simplification of having only two available tags.

As networks grow, when connections link together nodes, specific properties emerge, despite a certain degree of randomness. One common phenomenon, known as **preferential attachment**, defines situations where nodes prefer to build connections with *more popular* ones. This notion has been handy in understanding the emergence of large hubs and, overall, seeing how structures in different domains tend to converge to similar degree properties. Moreover, one would informally call this property a *rich-gets-richer* rule since it happens when the probabilities of new link formations are proportional to current popularity.

Speaking about hubs, we could also view these network-growing phenomena in terms of whether existing hubs tend to keep in the same proportions or otherwise. The following definition clarifies the idea.

Definition 2.2.3 (Assortative Network)**.** *A network is called* ***assortative*** *if any node of degree k tends to form connections with nodes of a proportionally similar degree. Such a network will be perfectly assortative if the process happens exactly between nodes of the same degree; on the other end, if links tend to exist mostly between nodes with different degrees, we say we are dealing with a disassortative network.*

2.3 Machine Learning for Graphs

In the previous section, we dedicated time to give a background to graph analysis and to formalize the abstract items we would work with throughout this thesis. With this in mind, we now focus

on the machine learning approaches proposed to analyze graph data. This section is based on the Graph Representation Learning book by Hamilton (2020).

Most of Artificial Intelligence's (AI) progress over the last decade has occurred thanks to the well-known subfield of Machine Learning (ML), yet we shall not confuse both terms. For the sake of practicality, we emphasize to the reader that AI is not solely defined in terms of *machines that implement, show learning abilities, or optimize themselves somehow*; instead, AI touches a plethora of knowledge fields that range from rationality, decision theory, sound reasoning, behavior, and thought processes (Russell and Norvig, 2009).

As for ML, this thesis will utilize its theory as an engine to acquire new findings from our graphs. When defining what it means for a machine or a computer to *learn* new knowledge, we do it by modeling our task with a *mathematical function* f to be optimized, for specific parameters. While f is under-specified, ML practitioners can make assumptions about designing any algorithm that optimizes it. Nowadays, one of the most fundamental requisites to any learning paradigm relies on providing a dataset $\mathcal{D}$ of examples, which will help the process.

At this point, ML approaches are divided into two paradigms: *supervised* and *unsupervised* learning. When the set of examples provides a *mapping* between inputs x for f and actual values $f(x)$, we say we have found a **supervised** task. For an ML practitioner, the goal, in this case, will be to come up with an approximated version of f, say $\hat{f}_\theta$, that optimizes, or *learns*, a set of parameters θ by using the labeled examples $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $y_i = f(x_i)$, $\forall\ 1 \leq i \leq n$. The set of values $\{y_i\}_{1 \leq i \leq n}$ can specify any kind of output; in particular, if the range of values it takes is discrete, we say we are working with a *classification* task and we often call the y_i's the *labels* of the dataset. On the other hand, if such a set's range spans continuous values (such as $\mathbb{R}$ or $(0, 1)$), we refer to the task as a *regression*.

Supervised learning comprises most *prediction* tasks where $\hat{f}_\theta$ could find how many people appear in a photo or guess the most likely next word while generating a coherent sentence. Most supervised datasets must be carefully labeled for a model to learn

from them. This is not a trivial task and often requires time, effort, and prior expertise: for instance, a group of linguists might give us better insights on how to identify hate speech on a corpus of Facebook posts. For this reason, **unsupervised** learning comes as a fundamental paradigm to learning from properties of the data $\mathcal{D} = \{x_i\}_{1 \leq i \leq n}$ alone. *Clustering*, the task of grouping data points from $\mathcal{D}$ according to similar attributes, is a common practice that has gained significant benefit, especially from *deep neural networks*.

In practice, we may find datasets that may not have been fully labeled; for instance, a *Kaggle*[3] competition might have collected a series of stock market checkpoints, indicating the price of *some* shares over time; hence, a crucial issue will be to figure out how a model *can generalize its knowledge* about stocks by *just learning from a few ones*. Machine learning has accommodated itself to these scenarios using learning pipelines that take a **semi-supervised** learning approach. Nowadays, it can be argued that a good ML model should be able to *transfer* the knowledge acquired from a supervised task to a setting where novel and unlabeled data is presented. This approach has very notably worked for computer vision and language understanding tasks, as deep models have been able to benefit from extensive data collections like IMAGENET (Russakovsky et al., 2015) and Wikipedia.

2.3.1 Common ML Tasks for Graphs

We present typical problems when a graph is a central structure to be processed by an ML algorithm. Due to their versatility, many tasks can be formulated as network tasks. Regarding large datasets, unknown relations, or dynamic settings, important properties stemming from local to global connectivity inspire how ML operates.

For the rest of this section, we assume that we have a graph $G = (V, E)$; We assume it to be simple for practical purposes, as everything we discuss can be trivially done for digraphs. There are three basic parts of G for whom, traditionally, previous work has been devoted to predicting *labels for nodes*, deducing (new or existing) *edges between nodes*,

[3] `https://kaggle.com`

and *classifying the whole graph* according to its structure. These three major approaches are often framed as supervised (or semi-supervised) tasks.

In **node classification**, we would like to learn a mapping $\hat{f} : V \to Y$ that assigns each node $v \in V$ a label $y \in Y$. Under this formulation, we can completely frame the challenge of labeling different types of users in a social network by certain criteria; for example, we might want to learn their political stand according to their recent activity and other accounts they follow.

To learn a robust model that classifies a graph's set of nodes, a common practice is to take a sufficiently large subset of nodes $V_{\text{train}} \subset V$ as the training set. The practitioner should know how to manage the training node's relations, as they are not just mere data points from the standard supervised learning formulation. Moreover, for complex networks, where different neighborhood structures form, it is often non-trivial to work with a significant representation of all present phenomena in V_{train}.

Sometimes, connections between nodes are missing from recovering a network. **Link** (or *relation*) **prediction** aims to learn such a problem, that is, a function $\hat{f} : V \times V \to [0, 1]$ that tells whether any pair of nodes have an edge between them or not. Any incomplete relationship between entities can be framed as a link prediction task: from recommending new accounts to follow on Instagram to completing the missing protein interactions in a cell. For a more thorough discussion on link prediction, refer to Section 2.4.

Machine learning can be extended to labeling networks; **graph classification** learns a mapping that identifies the whole structure. A training dataset, in this case, will include complete samples of graphs, which increases its complexity significantly compared with the previously mentioned tasks. In biochemistry, molecules are often represented by their components and chemical bonds, which define a network; to learn how molecules will mix up to produce novel medicine, one can train a model on a variety of medicine active components to be able to identify new arrangements that will turn out to be part of new drugs.

Before continuing towards the quest of *learning features from graphs*, we shall stop and look at specific computational complexity concerns. *How large of a portion of a network would affect or influence a given node?* Regardless of how this question is answered, as the number of edges increases in a *quadratic way*[4], network science also tells us most large graphs come with large sparsity, i.e., $|E| << |V|^2$. Hence, one shall avoid designing algorithms that strongly depend on computations over the graph as a whole.

To address this issue, most feature computations, including graph neural network loss functions (Section 2.3.4) employ **negative samples** that contrast ground-truths of data. It is, nevertheless, worth noting that the sampling strategy choice does significantly affect any learning method. One can experiment with multiple ways of doing so, which often take into account the difficulty or the variety of node samples one considers inside a node's neighborhood. Other concerns beyond node sampling include tasks where directed and multi-relational graphs are involved, where the direction of a link is important, and where connections, themselves, might induce biases if imbalances of relational sampling occur.

2.3.2 Engineering Graph Features into Vectors

The typical mathematical way to specify any network, $G = (V, E)$, is inadequate to formally frame most ML approaches, as they operate on *matrix*-like structures. A first, and often *naïve*, approach to *vectorize* a graph is to take its adjacency matrix A_G as a full representation. From each node's perspective, a row from A_G can be considered a *one-hot encoding* feature vector as a function of its neighborhood. If G comes with edge weights, these vectors could represent a probability distribution of graph connectivity if normalized appropriately.

Adjacency matrices are (often) NOT all we need for graph vector representations! Sometimes, it comes handy to specify a matrix of l-dimensional *node features* $X_G \in \mathbb{R}^{|V| \times l}$ where further context about nodes can be incorporated. This information could be any-

[4] which can be shown from the adjacency matrix

thing from structural facts about a node's further neighbors to external attributes that characterize every node (for instance, consider all personal data belonging to a single user on Facebook). A popular framework to learn node features will be discussed in Section 2.3.3 for structural information.

2.3.3 Node Embeddings

We now revisit *node embedding* techniques, which map vertices of a graph into a vector space. These kinds of techniques are related to word embeddings, which appear in natural language processing. Ideally, the geometry of the resulting projection should recover the relationships defined on the given graph.

Observe that there is no way to easily frame this task as supervised, as there is no requirement for labels to be specified. Nevertheless, we still have structural information from any given graph $G = (V, E)$. With this in mind, we can frame an *encoder-decoder* approach that will try first to project each node to a vector space and then recover the original graph. Thus, the encoder is generally a function $\text{ENC} : V \to \mathbb{R}^d$ that maps to a d-dimensional space. When the encoder is parameterized by little or no learnable variables, we say that a *shallow embedding* approach is being used. A suitable decoder shall give us some graph statistics from the node embeddings, which can help us evaluate the algorithm's performance. A typical decoder function will work on two embedded nodes $\text{DEC} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$, and will tell *how similar* are its inputs. Thus, the goal is to learn a function that minimizes the reconstruction objective

$$\text{DEC}(\text{ENC}(u), \text{ENC}(v)) = \text{DEC}(\mathbf{z}_u, \mathbf{z}_v) \approx \mathbf{S}[u, v] \tag{2.7}$$

where we assume $\mathbf{S}[\cdot, \cdot]$ is a similarity measure between nodes and $\mathbf{z}_u \in \mathbb{R}^d$. A common way to achieve this objective is to use *stochastic gradient descent* (Robbins and Monro, 1951)

to minimize the empirical reconstruction loss over a set of training nodes $\mathcal{D}$, as follows:

$$\mathcal{L} = \sum_{(u,v)\in\mathbb{D}} \ell\left(\text{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u, v]\right) \tag{2.8}$$

It turns out that this encoder-decoder scheme usually serves to describe most graph embedding algorithms. It suffices to define a similarity measure ($\mathbf{S}$), which is targeted by Equation 2.7, as well as a corresponding loss function. For the rest of this section, we focus on two similar methods, whose inspiration stems from reconstructing a conditional distribution on connected pairs of nodes ($p_G(\cdot \mid v)$ for any $v \in V$). They both form a basis for how the methodology (Chapter 3) is presented in this thesis. Other popular embedding methods can also be classified on whether they use a minimum squared error (MSE) loss, such as inner-product-similarity measures (Ahmed et al., 2013; Cao et al., 2015; Ou et al., 2016) or if they use a L_2-distance between embeddings to denote similarity and minimize the position of similar nodes in a vector space (Belkin and Niyogi, 2001).

`node2vec` and `metapath2vec`

The `node2vec` (Grover and Leskovec, 2016) algorithm is inspired by the popular `word2vec` (Mikolov et al., 2013) strategy for to embed words into a vector space. `node2vec` incorporates some stochastic assumptions by interpreting the decoder function as a conditional probability distribution of visiting a node u on a t-random walk that starts at node v, i.e., $p_G(u \mid v)$:

$$DEC(\mathbf{z}_v, \mathbf{z}_u) \equiv \frac{\exp \mathbf{z}_v^\top \mathbf{z}_v}{\sum_{v_k \in V} \exp \mathbf{z}_v^\top \mathbf{z}_k} \tag{2.9}$$

To optimize a loss function, we assume we are given a dataset $\mathcal{D}$ of node pairs that are sampled from the desired distribution $(v, u) \sim p_G(u \mid v)$. At this point, we note that optimizing any function in the whole dataset would result computationally expensive: especially since the denominator of Equation 2.10 requires $O(|D||V|)$ operations in the worst case. Thus, the algorithm turns to *negative sampling* to overcome this threshold.

According to Grover and Leskovec (2016), the loss function is given by

$$\mathcal{L} = \sum_{(v,u)\in\mathcal{D}} -\log\sigma(\mathbf{z}_v^\top \mathbf{z}_u) - \gamma\mathbb{E}_{v_n\sim p_n(V)}\left[\log\sigma(\mathbf{z}_v^\top \mathbf{z}_{v_n})\right] \tag{2.10}$$

where $Z_u = \sum_{v\in V}\exp f(n_i)\cdot f(u)$ is a normalizing term, whose expensive computation is often overcome via *negative sampling*.

The crucial part of the algorithm occurs during such sampling step, where the authors propose to follow a *biased random walk* approach, similar to the well-known *skip-gram* technique in NLP. This follows by computing the transition probability $P\left(c_i = v \mid c_{i-1} = u\right)$, where c_i indicates which node the algorithm visits at step i. Intuitively, this distribution is biased to visit only closed neighbors of a *root* node c_0; eventually, this procedure repeats per each node.

2.3.4 Graph Neural Networks

Deep neural networks have significantly impacted graph-related tasks as much as they do in many other data-driven fields. The present section will discuss the general methods used to formulate diverse ways of processing large networks and learning features using neural layers. Most of these functions are inspired by other successful approaches, such as *convolutional* methods, or from further study of network properties, such as the *spectral theory of graphs*. In all cases, the first significant challenge is to figure out how to encode a graph into an *initial* vector representation that enables any deep network to explore its properties further.

Recall that a simple graph $G = (V, E)$ can be represented in many ways using its adjacency matrix, which does not require a specific order of nodes, and hence, of columns and rows. Ideally, for any row or column permuted adjacency matrices of G, their resulting latent representation must be the same after applying any neural layer f. More formally, f should satisfy either **permutation invariance** ($f(\mathbf{PAP}^\top) = f(\mathbf{A})$) or **permutation equivariance** ($f(\mathbf{PAP}^\top) = \mathbf{P}f(\mathbf{A})$), for any permutation matrix $\mathbf{P}$ and adjacency

matrix $\mathbf{A}$. The major strategy to comply with this requirement relies on implementing a message-passing scheme on how neurons are updated, which motivates the most popular way of referring to graph neural networks.

Neural Message Passing. Consider a graph $G = (V, E)$, along with a matrix of node features $\mathbf{X} \in \mathbb{R}^{|V| \times l}$. A GNN will have a set of *hidden embeddings* $\mathbf{h}_v^t$ to be updated at time t per each node $v \in V$. Intuitively, they will acquire information from the closest to the furthest neighbors concerning each of the graph's nodes. The update can be framed as:

$$\mathbf{h}_v^{t+1} = \text{UPDATE}^t(\mathbf{h}_v^t, \text{AGGREGATE}^t(\{\mathbf{h}_v^t, \mid \forall u \in N(u)\}) \tag{2.11}$$

$$= \text{UPDATE}^t(\mathbf{h}_v^t, \mathbf{m}_{N(u)}^t) \tag{2.12}$$

where $\mathbf{m}_{N(u)}^t$ is the *message* that is generated from v's neighborhood. UPDATE and AGGREGATE are differentiable functions that are conveniently chosen as neural networks.

To enlighten the way UPDATE and AGGREGATE functions are usually implemented, here we introduce the basic formulation of a *message passing* GNN:

$$\mathbf{h}_v^{t+1} = \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_v^t + \mathbf{W}_{\text{neigh}} \sum_{v \in N(v)} \mathbf{h}_v^t + \mathbf{b}^{t+1}\right) \tag{2.13}$$

where $\mathbf{W}_{\text{self}}$ and $\mathbf{W}_{\text{neigh}}$ are trainable parameters, while σ is a non-linear function, usually either $\tanh$ or ReLU.

A small simplification to the neural message passing step (Equation 2.11) can be achieved by adding *self-loops* to all nodes. Hence, we aggregate information from the set $N(v) \bigcup \{u\}$ that includes the update operation implicitly:

$$\mathbf{h}_u^{t+1} = \text{AGGREGATE}\left(\mathbf{h}_v^t \mid \ \forall v \in N\left(v\right) \cup \{u\}\right) \tag{2.14}$$

Note that the update presented in Equation 2.11 accumulates all neighbors' values only by adding them up together. This can often be unstable, especially for large neigh-

Method	Equation
Set pooling	$\mathbf{m}_{N(u)} = \text{MLP}_\theta\left(\sum_{v \in N(u)} \text{MLP}_\phi(\mathbf{h}_v)\right)$
Janossy pooling	$\mathbf{m}_{N(u)} = \text{MLP}_\theta\left(\frac{1}{\lvert\Pi\rvert}\sum_{\pi \in \Pi} \rho_\theta(\mathbf{h}_{v_1}, \ldots, \mathbf{h}_{v_{\lvert N(u)\rvert}})_\pi\right)$
Neighborhood attention	$\mathbf{h}_{N(u)} = \sum_{v \in N(u)} \alpha_{u,v}\mathbf{h}_v$

Table 2.1 – Some of the various types of set aggregation methods, defined to exploit certain characteristics of large networks.

borhoods which may imply some difficulties while implementing any learning method. A successful way to normalize is given by Kipf and Welling (2016):

$$\mathbf{m}_{N(u)} = \sum_{v \in N(u)} \frac{\mathbf{h}_v}{|N(u)||N(v)|} \tag{2.15}$$

A **graph convolutional network** (GCN) arises from intuitively defining spectral convolutions on graphs by multiplying a signal by every node. The formulation presented by Kipf and Welling (2016) also employs symmetric normalization and self-loop updates. Its message-passing function is defined as:

$$\mathbf{h}_u^t = \sigma\left(\mathbf{W}^t \sum_{v \in N(u) \cup \{u\}} \frac{\mathbf{h}_v}{|N(u)||N(v)|}\right) \tag{2.16}$$

Equation 2.15 is not the only way to aggregate and normalize neighbor embeddings. This process is usually thought of as a set operation aggregation, where properties such as *permutation invariance* shall be kept. Table 2.1 summarizes three popular aggregation methods: first, a universal approximator of a *set pooling* operation has been shown possible (Zaheer et al., 2017) using a combination of deep fully-connected neural layers; then, *Janossy pooling* (Murphy et al., 2018) computes features from all possible node permutations; and finally, it is possible to assign an *attention* weight to neighbor nodes in order to give a ”influence” score while aggregating weights (Bahdanau et al., 2015).

In a *relational graph convolutional network* (RGCN), the aggregation function is augmented to add multiple relation types, which induces a transformation matrix per relation:

$$\mathbf{m}_{N(u)} = \sum_{\tau \in \mathcal{R}} \sum_{v \in N_\tau(u)} \frac{\mathbf{W}_\tau \mathbf{h}_v}{f_v(N(u), N(v))} \tag{2.17}$$

where f_n is a normalization function.

Attention mechanisms and feature concatenation can be used to leverage generalizations of edge features from the neighborhood. This is particularly useful when we would like to accompany neighborhood embeddings with those link features. For instance, we can re-define the aggregation function as follows:

$$\mathbf{m}_{N(u)} = \text{AGGREGATE}_{\text{base}} \left(\{ \mathbf{h}_v \oplus \mathbf{e}_{(u,\tau,v)} \mid \forall v \in N(u) \} \right) \tag{2.18}$$

In this case, $\mathbf{e}_{(u,\tau,v)}$ represents any arbitrary vector representing the (u, τ, v) edge.

So far, we have only dealt with embeddings at the node level, yet nothing prevents us from learning representation on a graph level. Any such operation, known as **graph pooling**, would be capable of being implemented directly on small subgraphs, an event to come up with a single edge embedding. There are two approaches to achieving this goal. First, we could frame this *aggregation* problem as learning a mapping from a *set of node embeddings* to a fixed vector representation. This function could be as naïve as taking the mean of a graph's node embeddings; nonetheless, in Vinyals et al. (2016), a combined module of LSTM-based approaches with attention mechanisms are provided, which has become a popular pooling method.

The aforementioned approaches fail to exploit any structural cues found in the graph. Topology can be leveraged by performing some clustering (otherwise known as *coarsening*) over the nodes and using such information to learn a function that maps a node with the likelihood it belongs to a particular cluster. Once we learn such mapping, a plausible alternative is to *iteratively* run a GNN on an adjacency matrix that encodes strength

between graph clusters, while *coarsened* graphs decrease in size, and information is accumulated at the final stage.

Graph Isomorphism

To conclude this section, we turn back to a fundamental graph problem whose interpretation provides intuition to derive better approaches to graph representation learning. Our characterization of graphs in Section 2.1 was incomplete, in a mathematical way, since no notion of *equality* was introduced. There is no straightforward to define such a notion since there can be multiple ways of writing an adjacency matrix for the same graph. In the literature, the related notion of **isomorphism** between graphs is used to refer to graphs G_1 and G_2 that are identical, except maybe, in the way nodes are ordered on their respective adjacency matrices. We can formalize this notion by considering adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$. We, then, say that these graphs are isomorphic *if and only if* there is a permutation matrix $\mathbf{P}$ such that $\mathbf{P}\mathbf{A}_1\mathbf{P}^\intercal = \mathbf{A}_2$.

Algorithm 2: Weisfieler-Lehman Algorithm

```
1  l_1 ← EMPTY_ARRAY(length=G_1.V.len);
2  l_2 ← EMPTY_ARRAY(length=G_2.V.len);
3  for i ∈ range(0, G_1.V.len) do
4  |  l_1[i] ← G_1.V[i].degree;
5  end
6  for i ∈ range(0, G_2.V.len) do
7  |  l_2[i] ← G_2.V[i].degree;
8  end
9  for i ∈ range(0, G_1.V.len) do
10 |  l_1[i] ← HASH(l_1[i], {{ l_1[i-1] }});
11 end
12 for i ∈ range(0, G_2.V.len) do
13 |  l_2[i] ← G_2.V[i].degree;
14 end
```

Transfer Learning on Graphs

While the *pre-training and finetuning* scheme seems to dominate most modern deep learning pipelines, implementing such benefits into GNNs is far from trivial. The simplest version of any training loss of a GNN (such as Equation 2.11) can be invariantly utilized to learn the ubiquitous complexity of an extensive network. It then makes sense to separate this process from any node or link classification task. Nevertheless, in Veličković et al. (2018), it is shown experimentally that a *randomly initialized* GNN would already incorporate the same representation power as anyone pretrained on a neighborhood reconstruction loss. This is also made evident in Hamilton et al. (2017), and Kipf and Welling (2016) that the connections one can draw from message passing to the Weisfeiler-Lehman algorithm (Section 2.3.4) explain this issue.

Nevertheless, the same authors in Veličković et al. (2018) introduce a method to pretrain a graph neural network, yet utilizing a different kind of loss. Such method is referred as *Deep Graph Infomax* jointly leverages information at node and graph levels. To do so, it is necessary to produce a "corrupted" version of the graph G to be optimized, which can be achieved by slightly modifying any node embedding or adjacency matrix in a stochastic way. All in all, further research has been proposing *unsupervised* ways of learning graphs. The crucial takeaway from them is that they coincide in optimizing mutual information at certain levels of representation or any "clever" negative sampling-based methodology.

2.4 Link Prediction

What does *finding a good video recommendation* has in common with *computing the best way to relate two concepts on a knowledge graph*? Both tasks can essentially be abstracted into predicting novel or existing links in a graph. Despite the diverse information large network, edges, links, or arcs can encode, recent deep learning techniques can be generalized well enough to diverse settings. For the current thesis, **link prediction** will mean that the like-

lihood of a certain activity that a social media user performs, given time; furthermore, the by-product of this task will be of great interest, especially considering the representational richness a deep model can abstract, as discussed in Section 2.1.

Graphs often come incomplete. When dealing with missing links, one can distinguish the situation when working with an entire graph is impossible, deleting existing edges and training a model to recover the original edge set. The former type of scenario is referred to as **inductive learning**, as presumably *knowledge will be generalized from existing information*, while the latter one is, in general, called **transductive learning**. Traditional supervised learning falls into the transductive category, while inductive approaches incorporate some lack of supervision, which requires working with unseen data points from train/test sets.

A complete simple graph $G = (V, E)$, with $n = |V|$ has $\frac{n(n-1)}{2}$ edges. Hence it is trivial to deduce that any link prediction will have time complexity $\Theta(n^2)$ in the worst case. On the other hand, most existing methods provide a probability score of attachment between every node.

Local Topological Predictors

The typical way of formalizing the link prediction task consists in providing a $score(u, v)$ for the likelihood of any pair of nodes $u, v \in V$ to be connected. This value can also be interpreted as a *similarity* measurement between u and v. Thus, every method will likely implement different notions of similarity between nodes; for instance, we may assume the *small world* hypothesis, which asserts nodes are strongly related by their path distances, to design a proportional score (Liben-Nowell and Kleinberg, 2003a). Next we provide some common ways of defining the aforementioned $score$ function.

People with many friends in common are likelier to be friends with each other. This social statement implies a naïve way to connect nodes in a graph:

$$\text{score}(u, v) = |N(u) \cap N(v)| \tag{2.19}$$

Jaccard Coefficient. This measurement, which stems from information retrieval, normalizes the aforementioned common neighbor score by all the neighbors each vertex has. While such interpretation might be convenient to relate to the actual equation, we note that the notion of a neighborhood can be extended beyond simple graphs, where *features* might be shared by nodes (multi-relational settings). In practice, we also add noise to the score to account for sparse networks or non-existent links.

$$\text{score}(u, v) = \frac{|N(u) \bigcap N(v)|}{|N(u) \bigcup N(v)|} + \text{Uniform}(0, \epsilon) \tag{2.20}$$

Degree Product (Preferential Attachment). Here we work under the assumption that the probability for a node u to have a new neighbor is proportional to k_u. This can be usefully put in the perspective of dynamic networks, where the number of nodes and edges increases over time. Hence, the probability of two nodes to become attached gives the score:

$$\text{score}(u, v) = k_u \cdot k_v + \text{Uniform}(0, \epsilon) \tag{2.21}$$

Adamic/Adar. The idea here is to measure how related are two given nodes. Once again, while this score is presented, in the simplest case, as a measurement of neighborhoods, this can be extended to multi-relational settings.

$$\text{score}(u, v) = \sum_{x \in N(u) \bigcap N(v)} = \frac{1}{\log |k_x|} \tag{2.22}$$

Katz Score. This score favors the attachment of any pair of nodes that have, overall, the shortest set of paths within each other.

$$\text{score}(u, v) = \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot |\text{paths}_{u,v}^{\langle \ell \rangle}| = \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot A^{\ell}[u, v] \tag{2.23}$$

PageRank

Page et al. (1999b) introduced the important *PageRank* algorithm, which remains a fundamental building block of understanding network structures and engineering algorithms that are scalable in practice. Essentially, this method of web page ranking, according to the original formulation gives us a method to figure out the most important nodes in a graph for a given one.

Consider running multiple *random walks* that start from a root node u. The average behavior and outcomes (final nodes) will, intuitively, summarize the graph's structure that follows upon u's neighbors. Moreover, this procedure shall give information on the proximity of u with any other node v. Note that there are no guarantees that any nonzero entry of any transition matrix P could lead to over-exploration of the network; in a very extreme case, this can account for very distant nodes which are not necessarily part of any *shortest path*. Hence, we introduce a parameter α to randomly *reset* to the initial exploration step.

$$\pi_u = \alpha P \pi_u + (1 - \alpha)\mathbf{e}_u \tag{2.24}$$

By repeatedly running this stochastic process, the authors derived a score function to effectively quantify how important is node v to u:

$$\text{score}(u, v) = (1 - \alpha) \sum_{x \in u \rightsquigarrow v} P[x]\alpha^{|x|} \tag{2.25}$$

where $P[x]$ is the probability of traveling across a walk $x = \langle v_i \rangle_{i=1}^{k}$ of, length k, i.e., $\sum_{i=1}^{k} \frac{1}{k_{v_i}}$.

2.4.1 Subgraphs, Embeddings, and Attributes for Link Prediction

Progress in *deep learning* (DL) has had an impact on link prediction, as well. Zhang and Chen (2018) present *SEAL*: a method that leverages (node) *attributed* graph neural networks to compute a set of edge features that will help to predict the existence (or not) of

any given link. The letters of this acronym stand for the important parts of the proposed methodology: local **subgraph** extraction to characterize each edge's surroundings and node **embeddings** as information about their **attributes**. Everything, then, is condensed to achieve a link **prediction** task.

The essence of the proposed framework by Zhang and Chen (2018) relies on a **node labeling** algorithm. This process would assign an integer label to every node to mark their different roles within their local neighborhoods. The intuition behind this is that labels characterize the different relative positions of each given pair of nodes, which imply distinct structural roles. The algorithm presented by the authors (*Double-Radius Node Labeling*) is inspired by the WL graph kernel algorithm (see Algorithm 2).

The previous feature construction approach has been designed to work well within a deep learning architecture. Competitive arguments and results have made use of Zhang et al. (2018)'s *Deep Graph Convolutional Neural Network* (DGCNN): a framework designed to extract useful features for general graph classification and to sequentially process the parts of a graph so that any other model could work with them. In particular, three computation stages are proposed: a collection of *graph convolutional layers* to extract local substructures, a `SortPooling` layer that sorts vertex features, and a combination of convolutional and fully-connected layers that read the sorted representations. Indeed, `SortPooling` is the previous work's main contribution.

A graph's order can be defined according to its application setting, regardless of the usual *node permutation invariance*: for instance, a text graph could be sorted in word dictionary order. For this thesis, we care about the *structural role* (i.e., the position) of each node within its local neighborhood, which is a concept that DGCNN captures through `SortPooling`. The idea is to pool information according to a *graph labeling* algorithm implemented upon the graph's topology. For such a goal, a version of the WL algorithm is typically used to group nodes with similar roles.

2.5 Misinformation Spread

Modern times correlate with *faster times*: news originates as quickly as ever as information spreads more easily thanks to the Internet. News outlets have evolved along with technological innovations that *connect* humans in different ways. While physical distances still separate communities and their culture, virtual links foster the need to acquire, process, and digest novel information. Determining the process of *misinformation production* is still an open question that could be addressed from different perspectives. An attempt to frame it, for instance, has been shown by Ruths (2019), in which five key production elements are identified: *publishers*, *authors*, *articles*, *audience*, and *rumors*.

In a densely connected society, malicious rumors spread faster than ever, as networks are not guaranteed to preserve trustworthiness. This type of activity, often dubbed **misinformation spread**, has been gaining popularity from different angles in the academic community. Computationally speaking, its importance has risen as a byproduct of the success of information technology, such as social media. While it can be argued that computer science has provided an *infrastructure* for misinformation spread, it still constitutes an active and essential area of research. Studying social media to learn about human behavior has proven to be an important direction from which social scientists can draw meaningful conclusions (Ruths and Pfeffer, 2014).

In this section, we attempt to disseminate some of the crucial parts of the misinformation phenomena that motivate the writing of this thesis. We first start with the items that *are spread* (fake news), to slowly transition to *who or where* do they originate (fake accounts, bots, and trolls). Finally, we land into a context where both of the mentioned parts interact and can be modeled by graphs.

2.5.1 Fake News

As discussed by Zhou and Zafarani (2020), there are various ways to define *fake news*; yet a broad definition must include any news articles that make false claims, regardless

of publication outlet. Depending on a specific research the question, one can assume whether several articles have been published by unreliable news outlets or not.

Popular social networks exhibit some weaknesses that enable automated trolls to spread *fake news*. To model and decide the trustworthiness of a given claim implies having a prior notion of ground truth: in particular, *knowledge bases* stand out as graph-based approaches, where fake news detection gets reduced to link prediction of verified truths (Ciampaglia et al., 2015; Zhou and Zafarani, 2020). Nevertheless, the graph structure of a given social media outlet has been part of recent work, thus, permitting to include propagation patterns along with controversial claims to be verified; for instance, in Ma et al. (2018), the authors perform a rumor classification task based on extracted *news cascade trees*, using deep learning techniques, such as *recurrent neural networks*.

Thanks to the great variety of attributes predefined by a social medium like Twitter, it is possible to customize many types of graphs as part of misinformation modeling and propagation. In Zhou and Zafarani (2019), the authors utilize network attributes to characterize spreading patterns of fake news: this is done by identifying news spreaders and quantifying the proportion of users that further, propagate claims in terms of social media distance and engagement with fake news.

Furthermore, Shu et al. (2019b) exploits social context to characterize fake news detection by proposing a *tri-relationship* joint embedding between publishing outlets, news articles, and users; link representation learning would characterize how each of the three parts interacts with each other. As a more general use of a social network structure, Shu et al. (2019a) construct *hierarchical propagation networks* that include news nodes, tweet and retweet nodes (which include a common news article), and reply nodes (derived conversations).

Fake Accounts

With the occurrence of important democratic events, such as elections, efforts have been put towards visualizing how bots' behavior trends over time (Yang et al., 2019). Among

the practical solutions to visualize how certain tweets and accounts *are polluting* the Internet, *Hoaxy*, a fact-checking interactive website, highlights the amount of features capable of extracting from a given Twitter account. The existence of a retweet network dataset, built from tweets spreading *hoaxes*, has also been reported (Hui et al., 2018).

2.6 Troll/Bot Detection

Exploring the vulnerabilities of social media to trolls and misinformation has drawn attention recently (Stewart et al., 2018), making special evidence on how political polarization allows *trolling accounts* to systematically take advantage of any scenario where global opinion consensus becomes jeopardized. The phenomenon of information *pollution* spread has gained attention from the community, as well; Lou et al. (2019) presents a multi-agent network model where bots interact with humans (both represented as nodes), and the former would employ different manipulation strategies to reflect on the latter.

Characterizing whether a Twitter account is controlled, either automatically or by a human, is a hard task; thus, recent related research utilizing machine learning techniques has permeated novel findings about malicious users and their *modus operandi*. For instance, in Mendoza et al. (2020), a bot detection framework is presented and benchmarked on a large data set of tweets: by getting access to a sufficiently large subgraph of interactions, the architecture first computes graph embeddings that will serve as features to jointly social interactions which, intuitively, group every similar node type (human or bot) together.

By investigating, formulating, and coming up with better ML models for tasks related to malicious online actors, we benefit not only from detecting those that are fully automated (bots) but also for partial automation (cyborgs) and, of course, trolls. This fact is also reflected extensively on Mendoza et al. (2020). With this kind of knowledge, we can provide ubiquitous applications, such as giving evidence of coordinated misinformation spread during the COVID-19 pandemic, as stated on Ferrara (2020).

2.6.1 Understanding Verified Trolls

Few recent works are related explicitly to the *Twitter Election Integrity* (TEI) datasets. In Zannettou et al. (2019c), the authors analyze *10M posts* identified as Russian and Iranian state-sponsored trolls. Attention is focused, especially during the 2016 US Elections, where they distinguish each troll's concrete political stand (e.g., Trump or anti-Trump supporters). The authors can also present a cross-platform *influence* model that quantifies, for instance, how likely events in a Twitter community influence subsequent ones within a Reddit community. With the help of *Hawkes Processes* (Laub et al., 2015; Zannettou et al., 2018), a stochastic type model that builds upon events that self-increase their repeating likelihood over time, the authors identify the *causal* event that made a URL propagate in different platforms.

Using a similar approach, in (Zannettou et al., 2019b), a comparison is presented between users with ties with the Russian *Internet Research Agency* and a random set of Twitter users; the authors find differences in terms of the content each group disseminate. Given the rich features provided within the TEI dataset (see Section 3.1 for further information), some works looked beyond tweet text and metadata. Furthermore, experiments that go beyond these attributes have been reported: Zannettou et al. (2019a) analyze 1.8M images from Russian trolls in the TEI dataset to conclude their posting activity matches that of real-world events. They further study how state-sponsored trolls select images and curate their posts toward a target user group.

In (Sharma et al., 2021), a troll classification task is conducted on a dataset collected from the Internet Research Agency (IRA) targeting US-related events. The authors use temporal point processes within a mixture density network to capture user behavior characteristics. This thesis differs from their work in the datasets and the learning framework employed. Specifically, we utilize two datasets distinct from the IRA to support our findings. Our non-troll control user crawling method is different, as we use available troll hashtags and user mentions. In contrast, their crawling method relies on a manually

specified list of keywords (see Section 2.7.1 for details). Additionally, their approach is based on temporal point processes, whereas we use a graph-based method here.

2.7 Sampling Social Media Datasets

Sampling from social media is a common yet not straightforward task. Here we review the related literature. Multiple works are concerned with efficiently capturing the underlying structure of a massive network while preserving large-scale observed properties, such as degree centrality, betweenness, or keeping the number of shortest paths between nodes (Leskovec and Faloutsos, 2006). The simplest sampling method for a graph can be based on multiple graph traversals, where observations can be thought to traverse neighbors from a starting root, i.e., implementing BFS or DFS. However, this can often induce degree biases dependent on high-degree nodes. Kurant et al. (2011b) have proposed a heuristic to un-bias BFS sampling methods, which is proved to work empirically on many topologies. On the other hand, Leskovec and Faloutsos (2006) investigate the differences between random walk and Monte Carlo methods for sampling real-world graphs. They concluded that the former was more efficient than the latter ones; this was attributed to their ability to capture certain measurements of huge real graphs (degree centrality, *beetweenness*, number of shortest paths) in a smaller sample. In (Wang et al., 2011), the aforementioned algorithms are applied to large-scale social media graphs while finding no satisfying results as they fail to preserve specific properties of full-scale graphs. The ability to compute good estimators of such graph properties, especially when it comes to degree distributions are a fundamental step for many sampling approaches (Wang et al., 2011; Zhang et al., 2015). More recently, Yousuf and Kim (2020) propose a sampling method based on estimating the degree and clustering coefficients of a graph, to achieve a sample size upper bound by 1% of nodes.

We don't need access to or sample the entire graph in many applications. Here the most commonly used method is *Snowball sampling*, which is a breadth-first search (BFS)

approach where samples are obtained by considering every sampled node's acquaintances. Although commonly used, this method might result in a bias towards sampling specific groups of nodes and become problematic in certain contexts (Browne, 2005). This motivates employing alternative probabilistic approaches that attempt to control the bias when choosing nodes and relations. For instance, Kurant et al. (2011a) develop a subgraph sampling approach where a graph's topology is estimated from a set of observed nodes along with some properties of themselves and their neighbors. These methods can be extended to graphs with multiple types of relationships. For example, Gjoka et al. (2011) combine multiple random walk runs, defined on every graph relation, to obtain a representative sample of a given multigraph.

2.7.1 Sampling Data from Twitter

Twitter data comes with diverse components, from which several approaches can be taken to reconstruct any interesting network. *Retweet* networks have been of special interest throughout relevant work (Ma et al., 2018; Sharma et al., 2021). While retweets give information on how user-to-user interactions occur, other features, like hashtags, would provide a list of topics relevant to each account. Recent work shows how crawling (and sampling) data leads to domain-specific methods (Addawood et al., 2019; Zannettou et al., 2019b), which are defined depending on a given task's requirements. Few prior works employ sampling for the specific task of studying the activity of trolls. Zannettou et al. (2019b)'s approach consists in sampling enough examples to match a given set of troll tweets so that the former's average distribution over time resembles the latter. It is worth mentioning that these trolls come from the same sources this work gathers its data for experimental purposes. See Section 3.1 for further information. Collecting two distinct classes of users, such as *trolls* and *non-trolls*, possesses another challenge. In the context of the 2016 U.S. Presidential election, Addawood et al. (2019); Badawy et al. (2018a,b); Luceri et al. (2020) have based their troll data mining on a dataset collected by

Crimson Hexagon, a former analytics company that provided paid access to such data[5]. For non-trolls, the authors used a manual list of keywords and hashtags, which were associated with 2016 U.S. Presidential candidates, equally represented, e.g., #donaldtrump, #trump2016, #hillaryclinton, #imwithher, and streamed related tweets for the same period the trolls were active. Furthermore, they also considered tweets from the users of these tweets, even when not including the election- related keywords, but only for those users who did not retweet trolls. According to the authors, this strategy would help to understand trolls vs. non-troll users better. While proper for content-based methods, this sampling technique is incompatible with graph-based methods. As engineered, one likely observes no interaction between the non-troll and troll accounts within this sampled data. In troll detection, for example, this could result in overestimation in our datasets where trolls interact and coordinate, whereas the control group users are disjointed.

In this thesis, we want to *sample graphs from Twitter data*. In particular, we take a network-based approach by reconstructing several graphs over time, which would be defined according to common user-to-user and user-to-hashtag relationships (Section 3.2). As crawled data often contains unnecessary context, it is crucial to implement adequate ways to sample graphs, which would (attempt to) resemble large-scale phenomena within a smaller community.

[5] `https://www.recode.net/2017/11/2/16598312/russia-twitter-trump-twitter-deactivated-handle-list`

Chapter 3

Methodology

3.1 Modelling Troll Activity with Graphs

3.1.1 The Twitter Information Operations Dataset

The Twitter *Information Operations* database has been constantly updated since late 2018. In line with their *transparency* objectives and intending to help the community to fight against *state-backed entities*, the social platform has been inviting members of governments and academia to further investigate, learn, and build technology using their archives. In October 2018, a set of $4,383$ accounts were released to kick-start the program[1]. The released activity was organized according to their alleged place of origin, in that case, either from the Russian *Internet Research Agency* (IRA) or Iran.

In subsequent releases, the practice of ordering any released data by place of origin was maintained: among other important implications, this allowed to get a better glimpse of why and how those users are operating, and what they are targeting. Every release

[1]Twitter's transparency website (Twitter, 2022) serves as the main source for every release's information and description. Most notably, every hashed archive can be easily accessed via the same website.

includes most of the metadata that can be extracted from querying the *Twitter Developer API*[2], except for the `userid` and `screen_name`, which are hashed for privacy issues[3].

All released users have already been suspended. Moreover, all releases include both, a list of users and their metadata accompanied by a list of tweets, also with metadata such as the number of likes and retweets received, along with the list of mentioned users and hashtags. Although their "lifetime" dates (from creation to suspension) are not specified, they can be inferred from the dates of their earliest and latest tweets, which gives a good approximation. For this project, we focus on the lists of mentioned users and hashtags, which provide crucial cues to each troll's interaction with their "outside world". Nevertheless, it is worth noting that every release has come with a *vast* collection of media (such as images and videos) that is part of the trolls' activity.

Having established this data collection's importance to understand *inauthentic influence campaigns* (Twitter, 2022), we now focus on automatically exploiting each release using ML models. According to Gadde and Roth (2018b); Roth (2020), no release has purely relied solely on *automated (bot) accounts*. Hence this can be considered a ubiquitous trace of activity generated directly by humans. In principle, to understand the behavior of these trolls, one would like to find patterns that separate these trolls from what *un-suspended, normal* users would ordinarily do. Hence, in this project, we provide the ingredients to address this issue by taking advantage of the nearest neighbors interacting with every troll.

3.2 Activity Collection Process and Data Augmentation

To work with the Twitter Election Integrity (TEI) data, we have built a set of scripts that download and handle their preprocessing. The data only includes the activity of *confirmed trolls*; thus, we needed to augment it with activity related to *non-troll users*. The

[2]https://developer.twitter.com/en

[3]Yet it is possible to request a version of their unhashed data, by filling out a form on their previously mentioned website.

counterpart of these trolls are the `user_mentions` they interact with, that is, their 1-hop neighborhood, which is the starting point of our **active** user control group. Even though some users do not exist anymore or have even been suspended, the number of *currently active* ones is significant enough for our purposes.

Fortunately, we were able to make use of Twitter's *Academic* API[4], to crawl a significant amount of activity and populate our datasets. These users provide examples of the targeted online conversations the coordinated trolls are built to target. To build a more diverse set of active users, we have considered the 2-hop user mentioned neighbors by trolls, which is obtained by looking at this relationship from the perspective of the 1-hop neighbors. We further detail the crawling process in Section 3.2.1. For simplicity, we would refer to a troll's set of active 1-hop neighbors as **closer** nodes; while **distant** nodes would be a troll's set of 2 (or further)-hop neighbors.

3.2.1 Sampling Active Users

In general, we would like that our trained models be able to leverage a troll's social network within its surroundings. Considering this, consider a list of troll activity (hashtags and user mentions) indexed at a particular time interval. Our goal would be to *expand* this list with more activity coming from non-suspended users.

We proceed by randomly sampling a proportion of mentioned users from the given list of troll activity. It is important to note that, at this stage, every user u keeps matched with the corresponding time interval u_t where their interaction with trolls has occurred. Observe that even trolls could be included in such a sample. Thus, we filter off all suspended users to obtain a list of *seed users* from where to pull more activity. For each one of these seed users u, we crawl any available data trace that falls inside u_t. This procedure's output constitutes our *closer* active user set, our first control group.

As hashtags on Twitter help users to easily engage in any (possibly, trending) conversation (Dorney, 2021), any troll interaction (or influence) with whichever active user

[4] `https://developer.twitter.com/en/products/twitter-api/academic-research`

		#senders	#receivers	#tweets	#hashtags	#user mentions
Russian	*Trolls*	129,877	1,428,207	1,581,542	455,853	972,354
	Closer users	43,630	895,790	899,625	228,754	667,036
	Distant users	4,738	972,041	985,591	545,773	426,268
IRA	*Trolls*	119,719	4,431,274	4,673,537	2	4,431,272
	Closer users	46,551	1,237,105	1,386,494	396,117	840,988
	Distant users	8,4058	6,910,592	6,999,959	2,993,350	3,917,242
Chinese	*Trolls*	38,698	448,298	613,496	211,013	237,285
	Closer users	43,230	1,924,381	1,951,513	587,044	1,337,337
	Distant users	2,590	215,274	218,177	102,448	112,826

Table 3.1 – Total number of *nodes* (senders, receivers), *links* (hashtags, user mentions), and *activity* (tweets) found either within the Twitter troll releases and their extracted closer and distant neighbors.

could occur without a direct mention. This fact is supported, especially, by events where trolls act in a coordinated manner to push specific ideas through hashtags (Gadde and Roth, 2018b; Roth, 2020). With this context in mind, we have added a *second level* of active users who allegedly engage in similar topics as trolls. These accounts can be seen as part of the distant neighborhood of trolls, which use identical common hashtags without direct interaction with the trolls while engaging within the same topics.

Table 3.1 summarizes the total number of users and hashtags involved in the obtained data. The number of senders corresponds to trolls reported originally inside TEI. Conversely, we compare such troll statistics to those from their crawled closer and distant total active users. Moreover, the number of receivers combines hashtags and user mentions, while the number of tweets also considers their repetitions to give the exact activity count. Given that we are retrieving data from the past, going back six years, the amount of activity we can collect is limited by the Twitter API and how much of the data is not removed by users over the years. However, we can still collect considerable activity for our control groups. It is also interesting to note that the obtained IRA trolls only use two

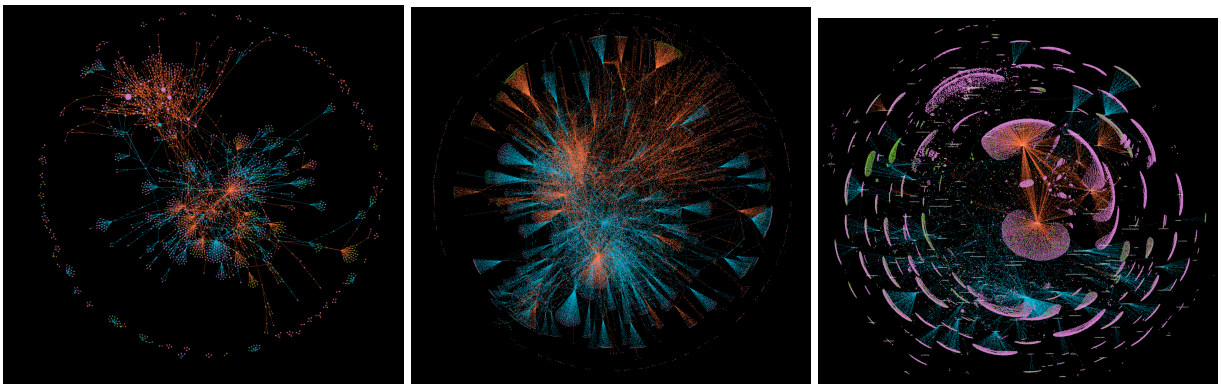


Figure 3.1 – Sample network visualization of the Russian (left), IRA (middle), and Chinese (right) datasets, on February 2017. Orange links correspond to troll activity, blue ones depict that from active users.

hashtags within their total activity. This impacts our active user crawling method in a unique way that we explain in Section 4.1.

3.2.2 Constructing Mention-Hashtag Graphs

To study the activity of trolls and active users, we first transform the data into graph snapshots. We provide visualizations across datasets and time to show how these datasets capture troll activities within their surroundings and with other control user groups. In particular, Figure 3.1 provides visualizations of mention-hashtag graphs constructed for each of our datasets for one month of activity, whereas Figure 3.2 shows the network evolution over one year for one of the datasets.

The accessibility to all closer neighbors within the trolls is not guaranteed, as they might mention other suspended accounts. In each dataset, we analyze multiple years of activity. For the moment, we focus on the technical challenges of using this data in a deep learning setting; nonetheless, we emphasize the importance and relevance of interpreting the diverse activity trends depicted in Figure 4.2, which will be discussed later in Section 3.3.3.

Constructing a single graph per dataset and processing it as a single training batch results in very impractical: as implied from Table 3.1, doing so would require over $1.5M$,

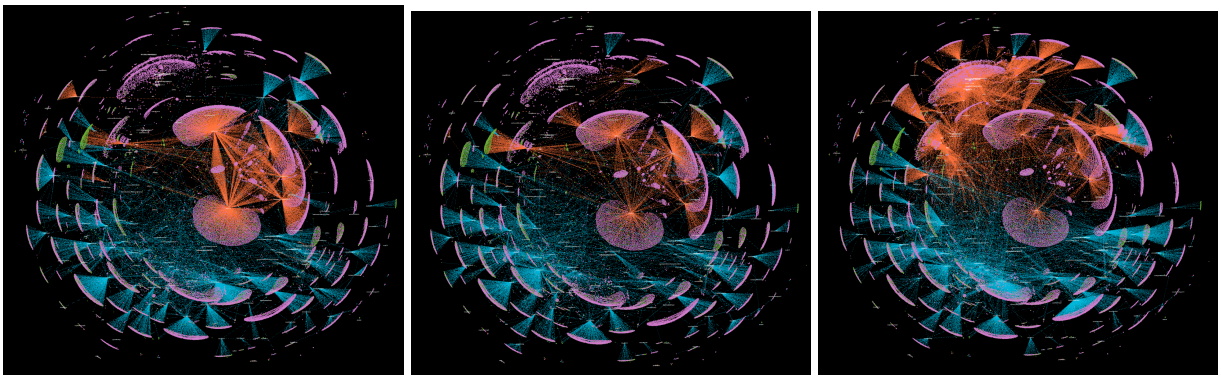


Figure 3.2 – Network development over 1 year in Chinese dataset. Every three visualizations correspond to four contiguous months, from left to right. Color coding is the same as Figure 3.1.

$4.5M$, and $2M$ feature only to represent Russian, IRA, and Chinese nodes, respectively. Therefore, we opt to work with individual chunks within contiguous timestamps so that any temporal aspect is ignored per chunk, as we prefer to capture all interactions happening at predefined time intervals, which will be controlled during experiments. Breaking a temporal graph into discrete time snapshots is a common practice Liben-Nowell and Kleinberg (2003b). On the other hand, activity peeks over time present a problem, as a set of few senders (Figure 3.1) are responsible for most activity (disconnected subgraphs, centered around a small number of senders, arise). Hence, we employ careful preprocessing to balance such activity, among other considerations explained in the following paragraphs.

From the proposed Twitter troll data, we propose to construct a *sequence of static snapshot* networks extracted from a set of tweets. For each one of those, an undirected graph $G = (V, E, W)$ is defined such that

$$V := \{u \mid u \text{ is a user}\} \cup \{h \mid h \text{ is a hashtag}\} \tag{3.1}$$

$$E := \{(u, h) \mid \text{user } u \text{ uses hashtag } h\} \cup \{(u, v) \mid \text{user } u \text{ mentions user } v\} \tag{3.2}$$

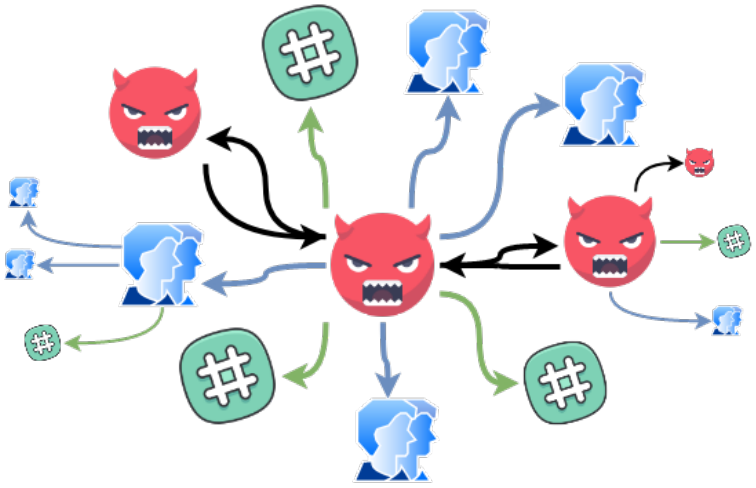


Figure 3.3 – This is the way we define links within our structural representation. Trolls (in red) are linked with the hashtags (in green) and other users they mention (in blue), which could also be other trolls. *Active* users are treated the same way.

Furthermore, $W : V \rightarrow 2^{\mathcal{H}}$ is a weighting function that assigns attributes for every node (Figure 3.3), where $\mathcal{H}$ denotes the dataset's set of hashtags. For our **mention-hashtag network**, W's role provides additional information to each user's close relationships, including neighboring hashtags, which are also abstracted as nodes. We, especially, distinguish the set of **senders** $\mathcal{S}$ (users that emit a tweet) from the set of **receivers** $\mathcal{R}$ (union of the set of mentioned users with the set of hashtags).

To accurately study the behavior of trolls using graph representation techniques, it is crucial to balance the activity between trolls and our control groups. This is because imbalanced activity can cause the model to overfit the majority class, which would hinder our ability to learn about the behavior of trolls. Additionally, the main goal of our study is not troll detection, which is a more complex task that requires addressing class imbalances within the model to be accurate. Below we explain how to extract graph snapshots with balanced activity from trolls and our control group. In Appendix B.1, we discuss this more and show that the overall patterns still hold in the unbalanced settings, and the conclusions remain the same.

In order to build a snapshot graph, $G_{\mathcal{D}}$, for a time interval $\mathcal{D} = (t_{min}, t_{max})$ we follow a procedure that may include sampling the number of receivers or links if any such quantity is out of balance with the number of senders:

1. We fix a time interval $\mathcal{D} = (t_{min}, t_{max})$ from which we extract all the tweets that were created no earlier than t_{min} and no later than t_{max}.

2. To correct *class imbalances* within each time interval, we examine the number of receiver users in the result from the previous step.

    - If $|\mathcal{R}|$ is *significantly* greater than $|\mathcal{S}|$, we opt to randomly take a sample of mentioned users and hashtags, whose number *roughly* matches that of the senders.
    - We, thus, *downsample* our chosen activity to balance the three types of nodes we are working with. This process helps to avoid biased predictions on certain classes.

3. To correct *activity imbalances* between time intervals, we examine the number of existing links between senders and receivers.

    - If the number of links, regardless of repeated mentions, exceeds a limit parameter ℓ_E, we randomly select a subset of links. We perform this action to balance the amount of activity in each snapshot.
    - Once again, this process helps us to control any undesired learned correlation on the final predictions.

4. Finally our (directed) adjacency matrix $A_{\mathcal{D}} \in \mathbb{R}^{(|S|+|R|)\times(|S|+|R|)}$ indicates whether a sender account mentions a receiver account and uses a certain hashtag.

It is essential to guarantee specific attributes for any chosen user to be in our *non-suspended* control group. We want to guarantee the validity of these accounts, which intuitively would imply the existence of a human that fully manages and makes decisions using the given *username*. On the other hand, we assume that in a 1-hop neighborhood

of mentioned users, the main topics covered by the given trolls are ”kept alive,” thus, depicting the intended influence targets to be enforced by coordinated organizations.

Among closer neighbors, we shortlist all those accounts that are suspended, as they inherently, may intersect the actual set of given trolls. The remaining active accounts constitute the complementary user behavior we want to distinguish. No further processing involving any exploration of these accounts' metadata is further involved. While these accounts might include some verified users, the important aspect between each other is that they have to be suspended but managed to remain on the Twitter network despite being targeted by trolls, even sometimes having interacted with them[5].

It is natural to observe that high sparsity levels will characterize smaller snapshots of activity concerning a predefined time frame. This is the sense of the number of available links, which might be orders of magnitude lower than the total number of possible links. This is not a true phenomenon for all available *small* snapshots, as clearly Figure 4.2 shows different peaks of activity happening at different times. Thus, we would like our model to learn any important pattern and, in principle, be able to generalize under different sparsity conditions.

An important aspect we shall observe is how the number of available tweets might influence any prediction. In principle, this variable **should not** be significant, regardless of how peaks of activities happen, as these events do not uniquely characterize any troll behavior. In contrast, we would like to capture the diverse ways users decide to involve and community within their close communities. For instance, consider a fixed set of accounts, both trolls and active users defining a mention graph as in Figure 3.3. If we fix the number of nodes for a given month while restricting the number of links to a shorter interval, say for ten days, we will end up with a vast proportion of inactive users (with degree zero) will be kept.

[5] `https://help.twitter.com/en/managing-your-account/suspended-twitter-accounts`

3.3 Analyzing Dynamic Mention-Hashtag Graphs

In this section, we first explain how to learn a representation for users based on their activity in a given time frame, captured by $G_{\mathcal{D}}$. For this purpose, we extract both type and neighborhood features for users to encode how similar users are based on how they are located within the overall graph (type), as well as their local neighborhood. We then explain how to use these representations for understanding troll activities. In particular, we discuss applying link prediction and node classification to predict behavior and type of users, respectively. This has implications for designing better troll detection applications. Finally, we discuss how clustering these representations help investigate the trolls' involvement in different discussion topics.

3.3.1 Representation Learning of User Activity

Extracting Type Features

Since our proposed graph construction delivers heterogeneous graphs where multiple types of nodes interact with each other; we utilize the `metapath2vec` algorithm which, recalling Section 2.3.3, biases every random walk according to predefined node paths. This algorithm is an extension of the very popular `node2vec`, which has been effective for many downstream tasks where a random walk would, in theory, be able to capture the most important features of a small neighborhood, as well as structural equivalence.

We consider four types of links defined by their incident nodes: `troll-uses-hashtag`, `troll-mentions-user`, `active-uses-hashtag`, and `active-mentions-user`. Given these links, we provide in Figure 3.4 a list of utilized (eight) metapaths between trolls and active users, as well as hashtags; moreover, Figure 3.5 shows the steps to get from tweets to type features using these metapaths.

Of particular interest is to capture and study interchanging activity between trolls and active users. Thus, in this case, the direct link assumption is dismissed for all paths connecting hashtags with any user. Such representation should also give a starting idea

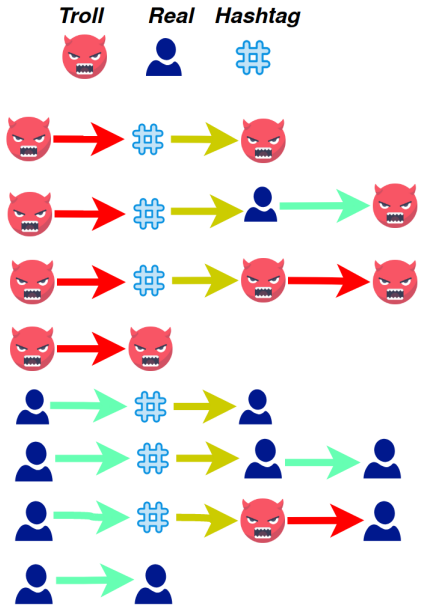


Figure 3.4 – Illustration of the chosen *metapaths* to bias our random walk-based node embedding extraction. Note that we have made explicit different ways of getting from trolls to active users, or vice versa, to be able to encode other (local) interactions.

of how users are naturally clustered into communities. We may find out sets of trolls targeting an agenda or sets of active users reacting to misinformation-like behavior.

Extracting Neighborhood Features

We now have defined a heterogeneous directed graph representation $G_{\mathcal{D}}$ for a time-definite set of activities $\mathcal{D}$; moreover, we have established how to construct its adjacency matrix $A_{\mathcal{D}}$ we augment with *negatively-sampled* links to produce a matrix $\tilde{A}_{\mathcal{D}}$. We now use $G_{\mathcal{D}}$'s structure to encode initial link characteristics according to their surroundings. For starters, we motivate the following process to account for various link prediction methods that have traditionally worked well (discussed in Section 2.4).

Most of these popular link prediction methods can be unified into an enclosing subgraph extraction algorithm (Zhang and Chen, 2018). These methods can be classified on the k-hop the neighborhood they use to compute a score. For instance, when $k = 1$, first-order methods, such as *preferential attachment*, only require making sense of the immediate

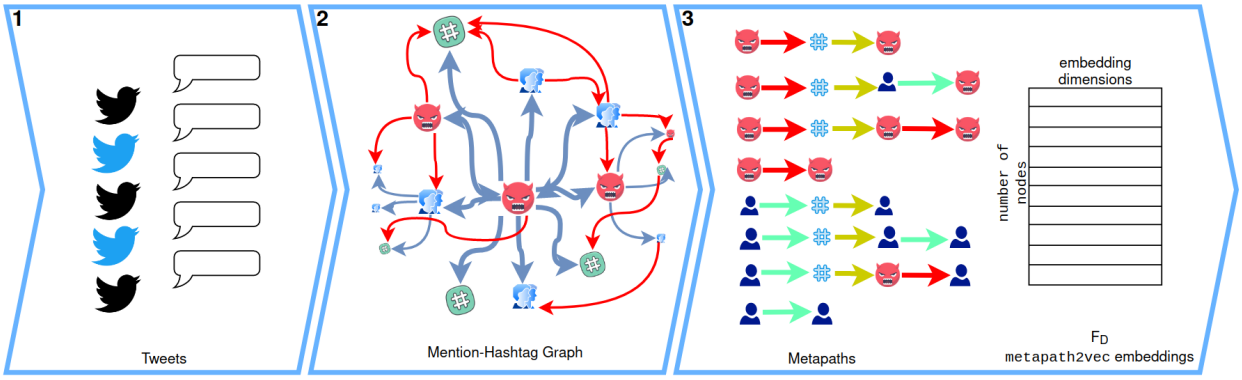


Figure 3.5 – Three steps to take a collection of tweets and produce *heterogeneous* structural representations. In this diagram, step 1 depicts our inputs, from which a *mention-hashtag* graph will be built on step 2. At this stage, we combine true links with *negatively-sampled* ones, displayed in blue and red, respectively. Finally, step 3 computes node embeddings using the `metapath2vec` algorithm, given a set of predefined metapath templates.

neighborhood of every node to make a verdict on the existence of any link. For second-order methods, such as the Adamic-Adar score (Adamic and Adar, 2003), a maximum of 2-hop information is used. This, in general, is extended to k-bounded neighborhoods, or even to the whole graph, for algorithms such as PageRank.

These heuristic-based algorithms have strong assumptions on how links form within a graph, such as the triadic closure phenomenon. However, they do not generalize to all types of graphs since networks originating from different domains follow distinct rules and patterns that determine the likelihood of every existing link. For this work, we take benefit of the heuristics mentioned above by leveraging the information obtained from different neighborhoods directly and allowing the model to find the most predictive patterns. The effectiveness of this approach is previously shown in Zhang and Chen (2018)'s **SEAL** (**S**ubgraphs, **E**mbeddings, and **A**ttributes for **L**ink Prediction) framework where these three types of features are incorporated via a *node labeling* algorithm that, intuitively, captures each node's role within its k-hop neighborhood. This model is trained with a link prediction objective, as explained below (see Section 2.4.1 for further information).

3.3.2 Activity Prediction and User Classification

Here, we discuss how to utilize link prediction and node classification tasks to monitor the similarity and differences in the activity of trolls and active users throughout time. After computing node type representations with the `metapath2vec` algorithm (Figure 3.5), we proceed to optimize a link prediction model that would learn from the different types of behavior given the original graphs and the learned type representations. The final embeddings from this link prediction model capture type and neighborhood features, and we use them directly to study how predictable the trolls behave. We also use the features extracted by this model to classify each node into trolls or active accounts. We must, as well, specify beforehand which *metapaths* to use (Figure 3.4) in case of using such an algorithm; moreover, we should predefine the value for the chosen k-hop neighborhoods as required to our link prediction model. These two hyper-parameters shape the performance of any training run, as will be further explored in Chapter 4. For the link prediction task, we essentially would like to predict the likelihood of the existence of a specific link in a graph $G_{\mathcal{D}}$, regardless of its significance. To achieve this, we rely on a negative sampling method that computes a random mask of false links, which does not depend on inherent characteristics such as degree distributions or connected components.

Once the list of predefined metapaths (Figure 3.4) is used to compute the node features $F_{\mathcal{D}}$ from $\tilde{A}_D$, we proceed with the SEAL link prediction pipeline (Zhang and Chen, 2018). This is the time when the node labeling algorithm takes place on the k-hop neighborhood of a given link $(u, v) \in E_{\mathcal{D}}$; even further, such information is augmented with each of u and v's type embeddings from $F_{\mathcal{D}}$. We produce a tensor $\tilde{X}_D$ of locally-labeled sub-neighbors, including negatively-sampled links. We are now ready to train a deep neural network – a DGCNN (see Section 2.4.1 for further information). We optimize the negative log-likelihood of the graph's predicted and existing links at training time. Algorithm 3 summarizes the link prediction process, including the loss computation, which is performed using the original adjacency matrix $A_{\mathcal{D}}$ and the one reconstructed from the

predicted links, namely $A_{\bar{Z}_{\mathcal{D}}}$. From Line 4, DGCNN provides both the vector representation of links and the reconstructed matrix.

Algorithm 3: Link Prediction Overview

Input: $G_{\mathcal{D}} = (V_{\mathcal{D}}, E_{\mathcal{D}})$, $A_{\mathcal{D}}$, k
Output: $\bar{Z}_{\mathcal{D}}$
1 $\tilde{A}_{\mathcal{D}} \leftarrow$ `negative_sampling(`$A_{\mathcal{D}}$`);`
2 $F_{\mathcal{D}} \leftarrow$ `metapath2vec(`$\tilde{A}_{\mathcal{D}}$`);`
3 $\tilde{X}_{\mathcal{D}} \leftarrow$ `links2subgraphs(`$F_{\mathcal{D}}$`,` $\tilde{A}_{\mathcal{D}}$`,` k`);`
4 $\bar{Z}_{\mathcal{D}}, A_{\bar{Z}_{\mathcal{D}}} \leftarrow$ `DGCNN(`$\tilde{X}_{\mathcal{D}}$`,` $\tilde{A}_{\mathcal{D}}$`);`
5 `lp_loss` $\leftarrow$ `neg_log_likelihood(`$A_{\bar{Z}_{\mathcal{D}}}$, $A_{\mathcal{D}}$`);`

Algorithm 4: Node Classification Overview

Input: $\tilde{Z}_{\mathcal{D}}$, $Y_{\mathcal{D}}$
Output: $\bar{Y}_{\mathcal{D}}$
1 $H \leftarrow$ `empty_tensor(dim=(`$|V_{\mathcal{D}}|$`,))`;
2 **for** $v \in V_{\mathcal{D}}$ **do**
3 **for** $u \in N(v)$ **do**
4 $H[v] \leftarrow$ `concat([`$H[v]$`,` $\bar{Z}_{\mathcal{D}}[(v,u)]$`])`;
5 **end**
6 $H[v] \leftarrow$ `mean_pool(`$H[v]$`)`;
7 **end**
8 $\bar{Y}_{\mathcal{D}} \leftarrow$ `mlp(`H`,` Y`,` `out_dim=2)`;
9 `nc_loss` $\leftarrow$ `neg_log_likelihood(`$\bar{Y}_{\mathcal{D}}$, $Y_{\mathcal{D}}$`);`

The previous process outputs a $|E_{\mathcal{D}}| \times j$ tensor of j-dimensional link embeddings $\bar{Z}_{\mathcal{D}}$ which we hypothesize to be rich enough to provide better insights on the actual differences of each type of user, whether troll or not. To ensemble this information, we concatenate together, in a matrix H, all the embeddings corresponding to every neighbor incident to any given node v. We then apply a mean pooling layer that squeezes down every tensor to a unique vector, which encodes node activity within the timestamp $\mathcal{D}$. With those activities encoded, we then apply a multi-layer perceptron layer to classify them, i.e., to obtain a binary output $\bar{Y}_D$; this last step is also fully trainable as it is optimized on the negative log-likelihood of true labels of nodes and their predictions. Algorithm 4 provides an overview of how this node classification works. This algorithm takes as input

the matrix of link embeddings computed by Algorithm 3 and uses a list of labeled nodes $Y_{\mathcal{D}}$ to define its corresponding loss.

3.3.3 Clustering User Activity

This section further explores the learned activity representations by clustering them. This allows us to understand better how trolls and active users interacted on various topics. This unsupervised analysis is practical in multiple domains, from natural language processing to computer vision (Mikolov et al., 2013).

Finding group of users that interacted with each other. Our training pipeline is optimized for link prediction. Hence nodes that cluster together should reveal similar activity traces. With this intuition, consider a mention-hashtag network $G_{\mathcal{D}}$ (as defined in Section 3.2.2), and its resulted embeddings $\tilde{Z}_{\mathcal{D}}$. This projection maps the input into a $\mathbb{R}^{\ell}$ space by taking model $\mathcal{M}$'s penultimate layer's output. To cluster $\tilde{Z}_{\mathcal{D}}$, we first apply t-SNE (t-distributed Stochastic Neighbor Embedding) (van der Maaten and Hinton, 2008), a powerful nonlinear dimensionality reduction technique to our link embeddings to obtain embeddings $\hat{Z}_{\mathcal{D}}$. After applying t-SNE, we use the KMeans clustering algorithm (Jin and Han, 2010) to identify an optimal number of clusters. The number of clusters was chosen based on the elbow method to minimize the average distance to cluster centers.

Finding the topics discussed in each group. To understand the content of each clustered activity, we look at the graph induced by the set of nodes in each cluster and retrieve the corresponding tweets. Recall that edges in our graph correspond to tweets mentioning another user or using a specific hashtag. More formally, for any cluster c, we restrict to only the tweets causing the corresponding links ($\top_{G_c}$). Then we use a language model to create their sentence embeddings: $\hat{\top}_{G_c}$. In particular, we have chosen to use the `RoBERTa`-based language model (Barbieri et al., 2020; Wolf et al., 2019), which has optimally been pretrained with around 58M tweets. We then cluster the obtained tweet

Symbol	*Description*
$G_{\mathcal{D}} = (V_{\mathcal{D}}, E_{\mathcal{D}})$	Any mention-hashtag network
$X_{G_{\mathcal{D}}}$	Node feature matrix of G
$\top_{G_{\mathcal{D}}}$	Tweets associated to every edge in G
$\mathcal{M}$	A link prediction model
$\mathcal{L}$	A language model
$\tilde{Z}_{\mathcal{D}}$	Link embeddings obtained from $\mathcal{M}$, given G as input
$\tilde{H}_{\mathcal{L}}^{\top_G}$	Text embeddings from model $\mathcal{L}$, given $\top_{G_{\mathcal{D}}}$ as input

Table 3.2 – A summary of all symbols used to represent any relevant quantity for our clustering assessment.

embeddings using different algorithms: UMAP and HDBSCAN. UMAP (Uniform Manifold Approximation and Projection) (McInnes and Healy, 2018) is a nonlinear dimensionality reduction algorithm that is particularly effective at preserving the global structure of data, while HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) (McInnes et al., 2017) is a density-based clustering algorithm that can handle noisy and high-dimensional data. This is a more accurate clustering method than the one used to find interacting users. We have picked a method for clustering our embeddings so that the cluster quality better corresponds with the embedding quality and could be used as a proxy to evaluate the embedding itself. Given evaluation of the language model used here is not within the goals of this thesis; we use the more complex clustering method in this case.

Overall, this lets us zoom into the diverse topics that constitute each clustered activity. To represent the topics, we use tf-idf (term frequency-inverse document frequency) (Havrlant and Kreinovich, 2017; Ramos, 2003) to obtain the essential words and phrases for each topic. Table 3.2 summarizes our clustering notation. Having these topics, we compare how trolls and our control users engaged in each topic when interacting, as reported in Section 4.4.

Chapter 4

Experiments

In this chapter, we first explain the datasets in Section 4.1, including how they are collected and sampled, as well as basic statistics on the size of the sampled data and how it changes through time. We continue, in Section 4.2 by reporting our representation learning results trained with a link prediction objective. We interpret the results of this section to compare the predictability of trolls and active users. These embeddings are then used to classify the users in Section 4.3. Finally, in Section 4.4, we take a deeper look at the embedding space, use clustering methods to activities around different topics, and contrast the engagement of troll and active users in those topics.

4.1 Experimental Data Preparation

We have collected some of the available tweet data from the Twitter Information Operations[1] database (as explained on Section 3.1). We have omitted media files as we do not use any of those in our methods. To automatize this collection process, we have built a *Python* script that uses the Google Cloud API where this *publicly available* information is stored[2]. Note that an authentication token from Twitter is needed to run this program successfully.

[1] `https://transparency.twitter.com/en/reports/information-operations.html`
[2] Refer to `https://github.com/alorozco53/deep-trolls` to access the tweet downloader code.

We have used *Twitter's Academic API*[3] to retrieve a portion of active users originally mentioned by trolls (details in Section 3.1.1). Our goal is to collect data on how trolls are engaging with non-troll users throughout a long period of time to be able to obtain rich (and fixed) network representations, where troll activity is made evident along with a set of non-troll users' (see Figure 3.3 for reference). Moreover, we would like to choose non-troll users involved with any of the three troll sources (Russia, IRA, and China) while justifying their use as contrasting control sets. For our purposes, we have checked that any chosen non-troll account *remains active* until its retrieval while having posted any tweets at intersecting times to any of the given trolls.

Our non-troll retrieval strategy depends on the specifics of each of our sources. Table 3.1 shows that for the IRA trolls, sampling users using the same hashtags would give incomplete information, given we only have 2 hashtags in this dataset. Overall, we have followed a series of steps that are visualized in Figure 4.1 to be able to sample our active users effectively. We split our non-troll sampling into three different strategies, from which we want to build a mention-hashtag graph (Figure 3.3) using activity from either *closer* (1-hop), or *distant* (2-hop, 3-hop) neighbors. For every dataset, it was possible to sample 1-hop mentioned users, as they were explicitly included as part of the TEI's metadata (4.1, top box).

Further analysis has been done to sample more distant users: we have sampled a proportion of the Chinese and Russian hashtags while keeping the time intervals in which they were utilized by trolls. This information gives us an indexed list of topic seeds where we can query Twitter's Academic API to find new active users. We follow this procedure to achieve our sampling goal, as depicted by the box in the middle of Figure 4.1. For the IRA trolls, a different procedure should be followed to overcome the scarcity of hashtags. We utilize *PageRank* (Page et al., 1999a) in a given mention-hashtag graph to rank each node and keep only the mentioned users whose score lies above the global average. Once again, we keep track of the time intervals these users are mentioned in the source datasets.

[3] `https://developer.twitter.com/en/products/twitter-api/academic-research`

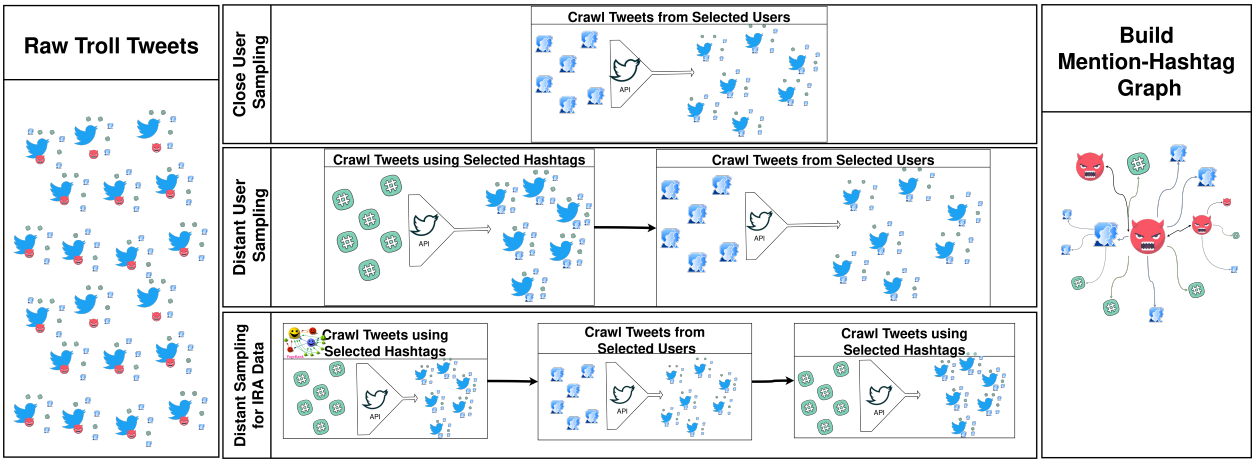


Figure 4.1 – This diagram depicts the various steps we employ to acquire non-troll users. At the beginning (leftmost box), a set of troll tweets is given as input. Then, the three middle boxes define each sampling strategy employed under this work: *closer active* users are sampled directly from the given tweets (top box); *distant active* users come after going through hashtag sampling (middle box); finally, the IRA dataset requires an additional non-troll PageRanking-sampling step (bottom box) to acquire 3-hop non-troll activity.

We then proceed by crawling tweets from these users, focusing on their used hashtags, to follow the same sampling approach explained previously (Figure 4.1, bottom box). It is worth mentioning that we have filtered out suspended users every time it was necessary to crawl from the API.

Our distant user sampling approach is similar to the ones explained in Section 2.7.1. For IRA sampling, instead of using a manual list of hashtags, as crawling seeds, we simply sample a set of hashtags without further restrictions to justify a more general approach to obtaining active users.

Figure 4.2 depicts how much tweet activity is produced by a troll and active users, independently. It is natural to see different peaks and declines of activity due to the different events the coordinated accounts were targeting. We have considered different time frames per each dataset based on the availability of data for active users.

From Figure 4.2, we also see no explicit guarantees to always have a balanced and dense dataset. Activity peaks from trolls may not coincide with those from active users,

yet any correspondence would derive from interesting behavioral patterns that will be discussed later. For practical purposes, this fact implies we shall look closely at the number of available nodes and links while performing any experiment, as degree imbalances may bias our results to any unwanted confounder variable.

To be precise, we refer as *senders* to those nodes who are producing tweets (only users, in this case), while we call *receivers* to those who have been mentioned by senders (the union of users and hashtags). Within Table 4.1, we observe that the number of receiver nodes is, on average, significantly dominant to the number of senders. Another important observation relies on the fact that, given our method, *the number of receivers is upper bounded by the total number of links* in every graph: for each (directed) link, every sender points to a hashtag or another user.

Moreover, in Figure 4.3, we see that the distribution of sender size against receivers varies across datasets – even further, across troll or non-troll sources. For instance, sampled graphs from the Russian troll dataset would have a steadier growth, in nodes and links, than the Chinese dataset, which in turn, would have a large number of 1-hop neighbors for a few nodes. By combining the information given by Figure 4.3 with the data statistics in Figure 4.2, we can see that

- the IRA is the dataset with the most mentioned users, while closer active links tend to grow faster in size than nodes;
- the Russian dataset has a steady growth with respect to each type of sender, which means that as the number of troll and active nodes grows, their activities increment proportionally, as well;
- an arbitrary Chinese troll would tend to mention more hashtags and users than any other user in the dataset.

We, henceforth, decide to take a similar-sized sub-sample of such groups under a predefined threshold; for most of our experiments, we have found it convenient to restrict

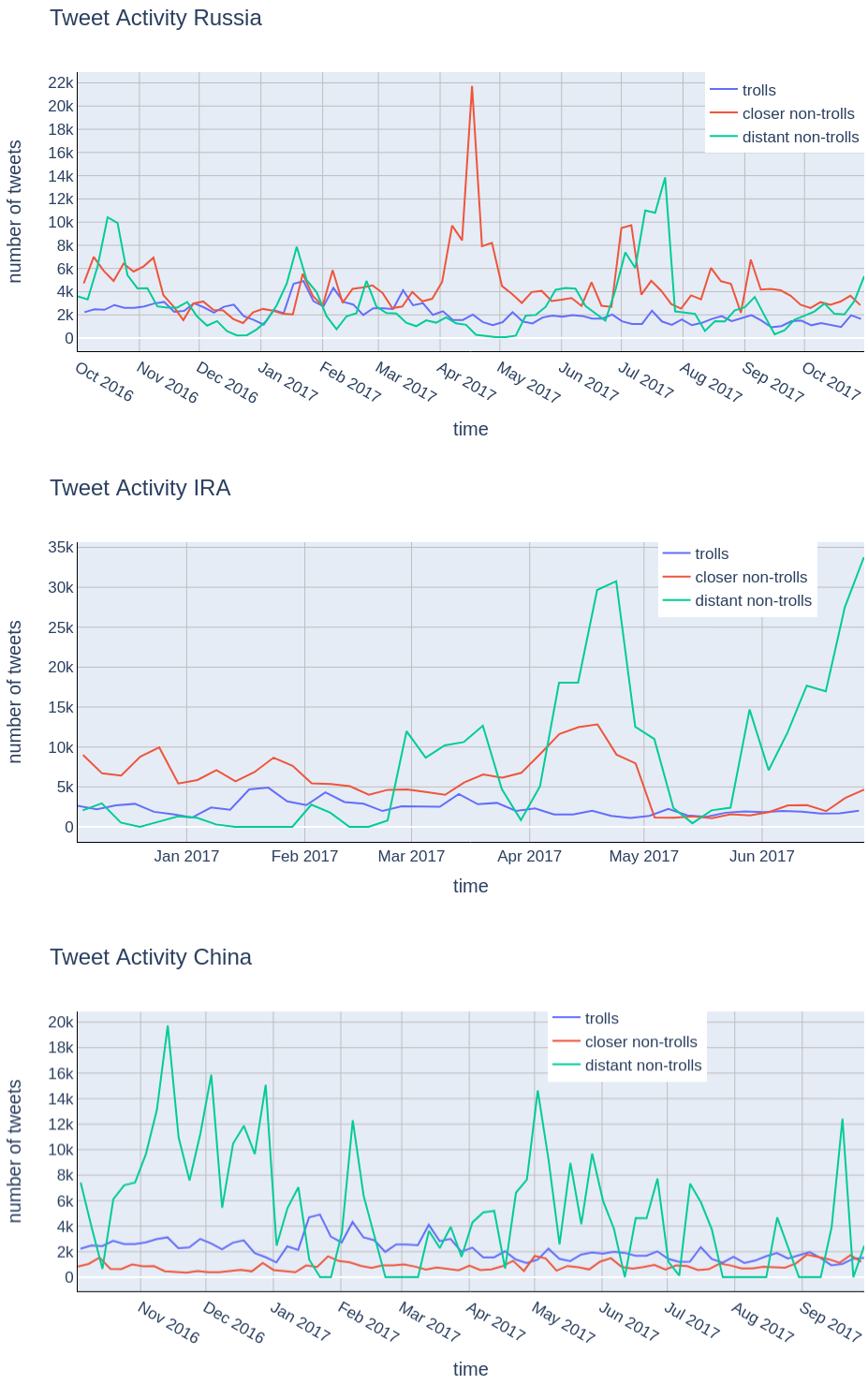


Figure 4.2 – Activity generated per time for the Russian (topmost), IRA (middle), and Chinese (bottom) datasets. Such activity is measured by the number of unique tweets generated per time unit. Troll tweets are depicted in blue, while active ones come in red.

Troll Senders	*Receiver*	`user_mention`	`hashtag`
Russian	84.90%	67.48%	24.97%
IRA	84.12%	79.71%	12.35%
Chinese	91.90%	72.87%	23.08%
Closer Active Senders			
Russian	98.97%	75.34%	24.69%
IRA	99.16%	72.46%	27.54%
Chinese	99.26%	76.87%	23.13%
Distant Active Senders			
Russian	99.43%	61.48%	38.52%
IRA	95.68%	70.43%	29.57%
Chinese	97.42%	63.57%	36.43%

Table 4.1 – Out of the total nodes in our graphs, here we compute the percentage of them that correspond to receivers (hashtags and mentioned users).

the number of total links to $\mathbf{5,000}$, with equal proportions of user mentions and hashtags. Moreover, we have truncated the number of receivers to roughly the same number of senders. The formerly mentioned types of nodes are the critical centers of every network, regardless of their troll nature. Thus, this process will not skip valuable information for optimizing any model. Refer to Appendix A for other data statistics.

4.2 Predicting Behaviour through Link Prediction

We now focus on predicting the existence of links on a given graph, where we do not distinguish between those generated by trolls or active users. This is equivalent to modeling the network's structure as a whole, as predicting the set of links fully defines a graph. To account for how any graph could be constructed over time, we perform experiments using different time intervals: 5, 10, 30, and 60 days.

For starters, we look into our pipeline's ability to predict the existence of links. Table 4.2 exposes our results averaged over five different training-testing splits: each row gives macro-F1 scores on the link prediction task on graphs originating at the indicated place

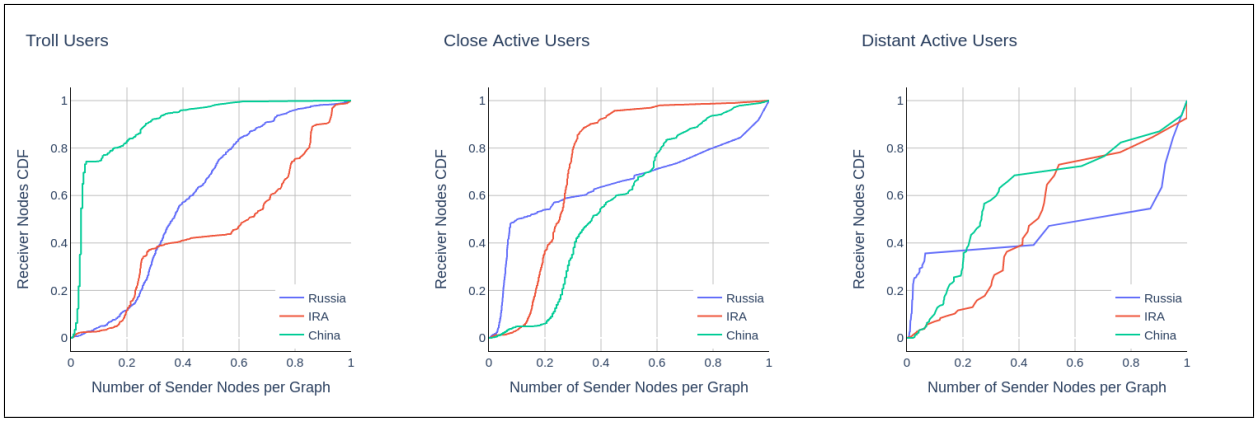


Figure 4.3 – Cumulative distribution functions (CDF) of receiver nodes, given the number of sender nodes. We have normalized each axis to roughly compare how many links a graph would have as its senders grow. These frequencies have been computed by counting all activity in 5-day intervals, the minimum time we use for any experiment (see Section 4.2) for further details. Hence, the x-axis represents the percentage of the number of nodes for each graph, while the y-axis distributes the proportion of respective receiver nodes.

of birth. Our results point out that including distant active users, *slightly* decreases overall performance in most settings while considering both sets of active users benefits the experiments' outcome.

Duration of the Time Interval

Sampling from ample time intervals means we could include troll-generated content that spans multiple activity peaks in our static mention-hashtag graph. This fact is easily observed from Figure 4.2 by considering sampling, for instance, 30-day tweets from the IRA dataset, ranging from June to August 2015. Moreover, expanding the duration of a time can also be seen as enriching any user's set of features; hence, we have performed multiple runs at different intervals to observe any benefits.

Table 4.2 shows a decreasing pattern for the Russian data, as F1 scores decrease from 90% to 82% when these trolls are contrasted with both closer and distant active users. This falling pattern is also observed when training with the two other datasets. When going

Link Prediction F1-Scores					
Active set	Dataset	5 days	10 days	30 days	60 days
closer	**Russian**	0.88 ± 0.027	0.86 ± 0.027	0.8 ± 0.029	0.77 ± 0.032
	IRA	0.88 ± 0.039	0.88 ± 0.038	0.87 ± 0.035	0.85 ± 0.037
	Chinese	0.94 ± 0.039	0.94 ± 0.037	0.93 ± 0.034	0.92 ± 0.036
distant	**Russian**	0.88 ± 0.040	0.88 ± 0.038	0.82 ± 0.027	0.77 ± 0.051
	IRA	0.82 ± 0.087	0.80 ± 0.110	0.73 ± 0.150	0.70 ± 0.160
	Chinese	0.89 ± 0.044	0.90 ± 0.050	0.87 ± 0.053	0.85 ± 0.050
both	**Russian**	0.90 ± 0.025	0.89 ± 0.022	0.84 ± 0.021	0.82 ± 0.023
	IRA	0.84 ± 0.079	0.81 ± 0.096	0.78 ± 0.100	0.73 ± 0.130
	Chinese	0.92 ± 0.034	0.92 ± 0.034	0.91 ± 0.036	0.88 ± 0.046

Table 4.2 – Link prediction performance scores. We report F1 scores for each experiment performed over 5, 10, and 60-day intervals. Additionally, we compare two model configurations that vary only in their use of the node labeling technique outlined in (Zhang and Chen, 2018).

from 5 to 10 days, only the Chinese dataset shows an improvement, only when experimenting with trolls against distant non-troll users. Therefore, our pipeline seems to work better with 5-day time interval splits: this criterion will be used for further experimentation.

Effect of Node Labeling

When introducing, in Section 3.3.1, our training pipeline, we mentioned that the *node labeling* algorithm, which is part of the SEAL framework (Zhang and Chen, 2018), would help us leverage each user's role to justify the existence of a link (activity point). Henceforth, we provide some ablations to verify its utility. Table 4.3 provides an analogous set of scores to those from Table 4.2, yet we have dismissed the node labeling algorithm for the former mentioned.

The node labeling algorithm's benefit is mainly seen in 5 and 10-day internal experiments. Notably, the IRA experiments get increases in F1 scores, going from 77% to 84%, when running on closer and distant active users. We also observe instances where this algorithm proves beneficial over more extended periods. This suggests that even when

Link Prediction *Node Labeling* Ablation					
Active set	Dataset	5 days	10 days	30 days	60 days
closer	**Russian**	0.87 ± 0.030	0.85 ± 0.029	0.80 ± 0.030	0.77 ± 0.024
	IRA	0.83 ± 0.066	0.83 ± 0.059	0.82 ± 0.038	0.83 ± 0.034
	Chinese	0.94 ± 0.060	0.92 ± 0.067	0.91 ± 0.057	0.92 ± 0.052
distant	**Russian**	0.86 ± 0.082	0.86 ± 0.061	0.81 ± 0.046	0.76 ± 0.046
	Chinese	0.87 ± 0.055	0.90 ± 0.051	0.86 ± 0.056	0.84 ± 0.058
	IRA	0.77 ± 0.096	0.75 ± 0.100	0.73 ± 0.110	0.71 ± 0.130
both	**Russian**	0.86 ± 0.069	0.86 ± 0.055	0.81 ± 0.045	0.76 ± 0.045
	IRA	0.77 ± 0.091	0.75 ± 0.098	0.74 ± 0.099	0.72 ± 0.110
	Chinese	0.88 ± 0.053	0.90 ± 0.050	0.86 ± 0.055	0.84 ± 0.058

Table 4.3 – Link prediction performance scores, on ablation without node labeling (Zhang and Chen, 2018). We report F1 scores for each experiment performed over 5, 10, and 60-day intervals.

the sampled link neighborhood becomes more complex, certain users still play a critical role in our objectives.

Effect of Active Users Set

Our three datasets include various types of users, while links can also be categorized according to where they originate. Active users, i.e., non-trolls, can be directly part of any troll's targets or be found on similar conversation topics (by crawling 2-hop users from hashtags). We, therefore, have decided to control for the use of a *closer*, a *distant*, or a mix of *both* neighborhoods, as depicted on Figure 4.5.

We have found that Russian experiments benefit from adding distant active users as, for instance, in a 5-day interval, we see an increase from 88% to 90% F1 score. This is not the case for both, IRA and Chinese experiments, in the majority of the results. Adding distant users seems detrimental for the Chinese dataset, even though both sets of active users imply an increase in F1 scores: within a 5-day interval, scores go from 94% to 89% and finally increase again towards 92%.

Arguing about the optimal way to split the data according to a time interval is somewhat nuanced. For the Chinese-based dataset, F1 scores support taking a 10-day interval;

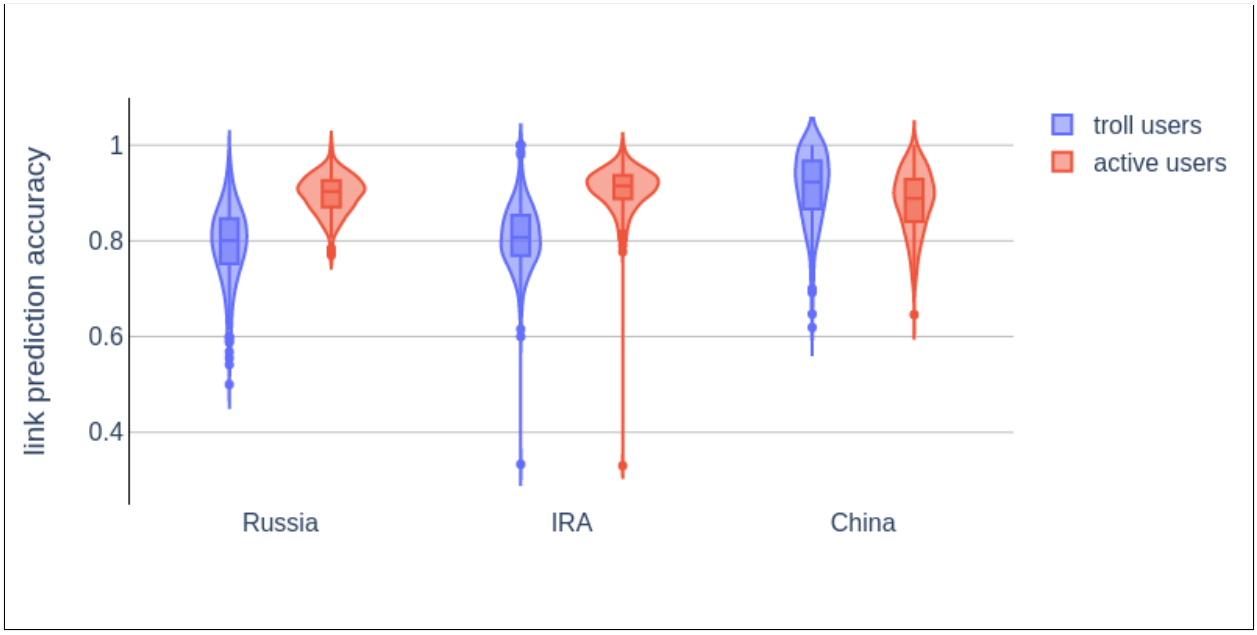


Figure 4.4 – The proportion of correctly predicted links per each place of origin, further divided by whether each activity was produced by a troll or by an active user. While Chinese troll predictability exceeds that for their corresponding active users, there is a vast amount of outliers which can explain the diversity of online coordinated strategies.

conversely, the Russian and IRA datasets perform best within a 5-day split. While larger intervals may signify increased available data and better insights for a model to learn, these results might suggest that excessive information does not always benefit our link prediction task. We have mainly focused on F1 scores to make any conclusions about our results, as we would like to dismiss all kinds of false positives.

Our task design would not directly say how likely it is to learn any information, particularly to every kind of user involved in the graph. Nevertheless, we can still measure the proportion of correct prediction per user type to account for *behavior predictability*. We report scores split by place of origin, indicating the proportion of accurate predictions by either trolls or active users.

Figure 4.4 shows such results; in this case, we average all correct predictions for a 10-day interval split using the metapath2vec and node tagging algorithms as optimal setup. For the Russian and Chinese-based networks, troll activity seems to be more predictable;

nevertheless, it is worth noting that the discrepancy with the active users is almost negligible, and these outcomes provide another insight into how well, does the proposed pipeline performs in general. Finally, note that these LP scores can be further broken down concerning the choice of closer, distant, or both active set. Figures B.2, B.3, and B.4, in Appendix B, present these accuracies by comparing the distribution of correct classifications, which, once again, correspond to the information given in Table 4.2.

4.3 Classifying Users through Node Classification

As part of our link prediction pipeline, we have learned embeddings for the given datasets, which can be seen as part of online behavior originating in two ways: either they come from trolls or active users. Testing how good are these vector representations to distinguish the classes mentioned above is an effective way to evaluate the goals set for this thesis. At this point, we would like to stress the fact that our studied environment constitutes examples of information exchange between suspended and non-suspended accounts: while active counterparts could be found within different types of niches online, we only focus on immediate scenarios where explicitly coordinated actions take place to change those from active users, potentially.

Figure 4.4 similarly shows our node classification scores as we did for link prediction. Every single experiment would include training loops on both link prediction and node classification: we first optimize the formerly mentioned task for a certain number of epochs to later train the piece of our pipeline dedicated for user classification, as pointed out in Algorithm 4.

Mainly focusing on F1 scores, we observe that the Russian and the IRA users benefit from larger day interval splits this time. In this case, the amount of necessary information for the model to succeed seems like a crucial confounder not solved by the sole link behavior learning performed before.

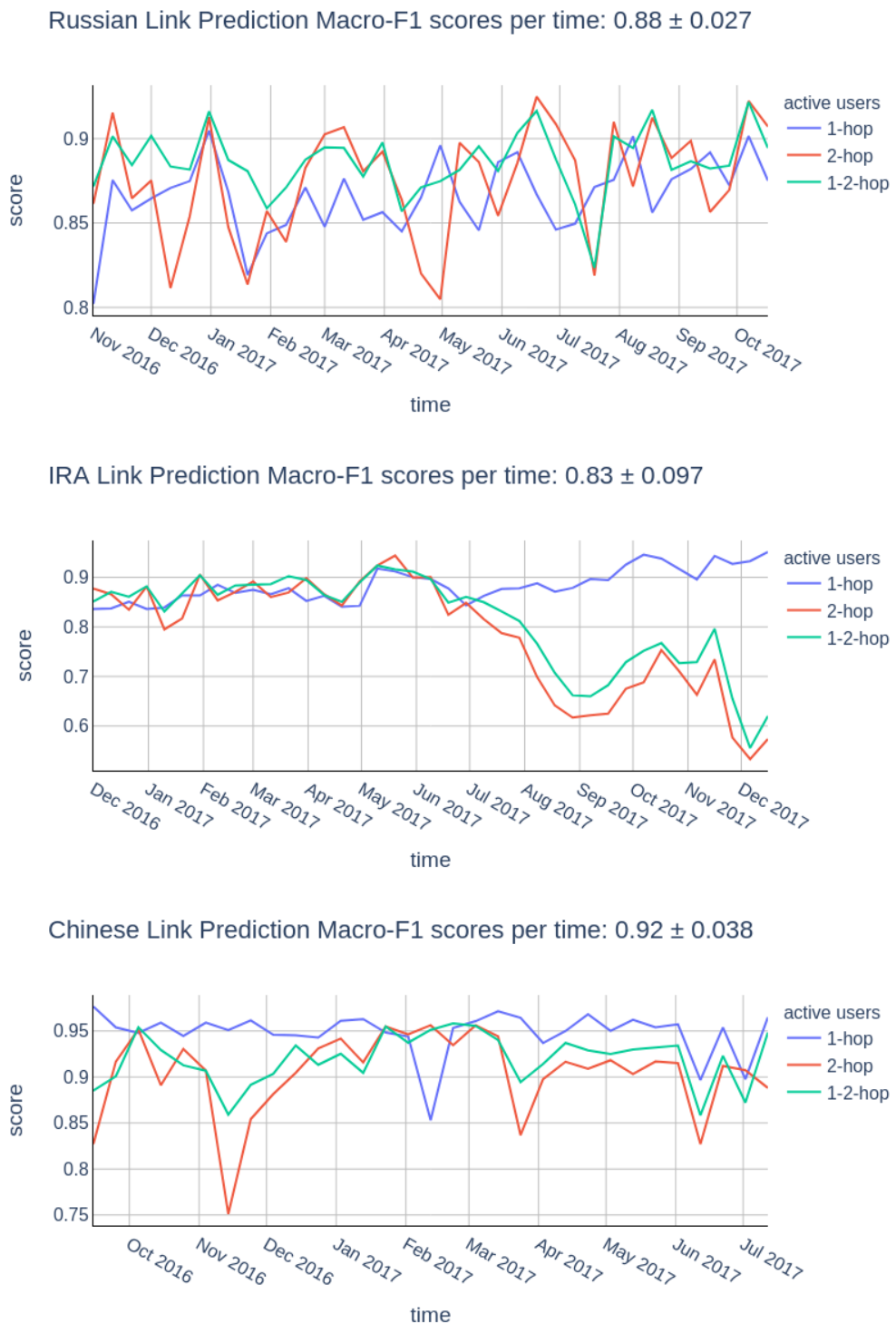


Figure 4.5 – Link prediction F1 scores per time, per data, according to the trolls' indicated place of origin.

Node Classification F1 Scores					
Active set	Dataset	5 days	10 days	30 days	60 days
closer	**Russian**	0.52 ± 0.24	0.43 ± 0.10	0.46 ± 0.15	0.43 ± 0.04
	IRA	0.43 ± 0.17	0.43 ± 0.14	0.40 ± 0.11	0.42 ± 0.08
	Chinese	0.69 ± 0.34	0.61 ± 0.28	0.61 ± 0.26	0.58 ± 0.25
distant	**Russian**	0.48 ± 0.30	0.43 ± 0.19	0.41 ± 0.15	0.39 ± 0.07
	IRA	0.45 ± 0.18	0.44 ± 0.14	0.43 ± 0.11	0.41 ± 0.02
	Chinese	0.87 ± 0.29	0.64 ± 0.30	0.61 ± 0.26	0.50 ± 0.19
both	**Russian**	0.53 ± 0.22	0.47 ± 0.16	0.49 ± 0.16	0.43 ± 0.03
	IRA	0.49 ± 0.19	0.48 ± 0.16	0.47 ± 0.12	0.44 ± 0.03
	Chinese	0.72 ± 0.29	0.65 ± 0.27	0.65 ± 0.26	0.50 ± 0.08

Table 4.4 – Macro F1 Node classification performance scores.

Following on the official reports of the Chinese release[4], these troll accounts would rather take a *"spammy"* approach in their operations; hence we would expect them to follow certain clear patterns that should identify them, although their differences from *bots* are still kept, as they also were suspended for diverse reasons, e.g., the sole fact of pretending to be other people. In contrast to what the IRA and Russian trolls usually target, these Chinese accounts were allegedly created to *cause political discord* in Hong Kong, specifically, by amplifying some messages related to protests. On the other hand, the Russian and IRA trolls were both targeting global events and had a large history of coordinated activities, dating since 2019; hence, we expect to find differences between them and the Chinese accounts.

The F1 scores we present in this thesis can be compared directly with other baselines. However, we are unaware of actual work on the three datasets we are considering here. Sharma et al. (2021)'s best approach reaches a 77% F1 score using a combined the IRA dataset, with the active ground-truths presented in Luceri et al. (2020), on the supervised classification task. In our case, we do not outperform these results for our IRA data; nevertheless, we achieve a promising outcome by obtaining 87% F1 score in our Chinese dataset.

[4] `https://blog.twitter.com/en_us/topics/company/2019/information_operations_directed_at_Hong_Kong`

As performances seem to vary between places of origin, even when not following a concrete pattern across different time intervals, it comes down to exploring the nature and reasons of the current dataset's trolls to understand what is happening. Differently from related literature, we repeat experiments over contiguous snapshots that allow us to examine troll activity within information peaks. This is reflected in F1 scores, as there are explicit situations where a lack of activity undermines performance. Furthermore, how we evaluate our model against active accounts is very different from related work (Luceri et al., 2020; Sharma et al., 2021), which distinguishes our IRA dataset from others. At the bottom line, discrepancies between proposed troll-active datasets imply that there is still no free lunch on the algorithm design for this task!

4.4 Experimental Interpretations

This section provides several mechanisms to interpret what happens inside our proposed training pipeline. In particular, we are interested in exploring how *posting tweets* took place at time intervals when relevant events occurred. Our pipeline freezes graphs that, naturally, possess a temporal component to learn their features. This procedure (Section 3.2.1) gets rid of any signal that would indicate user activity dynamics at training time; nevertheless, it is still possible to study the model's sensitivity to various input sizes. We measure this using *Pearson's correlations* between the amount of data and given scores per time.

Correlations with Graph Features

Table 4.5 presents the *Pearson's correlation coefficient* between all link prediction accuracy (over time) and three graphs (size) features: number of *trolls* (senders), number of *active* users (senders), and number of *receiver nodes*. The latter quantity is proportional to the number of links in any constructed graph, as our proposed method implies that the edge

set forms a *one-to-one* mapping from senders to a group of receivers (recall Section 3.2 for more details).

At this point, we focus on unveiling which network features influence the obtained results most. To answer this question, we can look into the correlation coefficients closest to -1 or 1. For the Russian experiments, we find the highest coefficients as we compare our results to the number of active users; in particular, this is stressed especially for distant users, which implies that this control group gives the most significant information for the model to reconstruct the actual Twitter interaction graph, according to how we have defined our link prediction task.

The latter conclusion can be similarly applied to other features: for Chinese experiments, receiver nodes (mentioned users and hashtags) would have the most important for the model's success, especially when distant active users are in play. We have, therefore, derived these implications by observing the coefficients above 0.3 (or below -0.3); furthermore, more interesting facts can be extracted from Table 4.5. For instance, the combined use of both closer and distant active sets is more beneficial as non-troll control groups. Providing the model with activity directly from troll interactions would include users not necessarily engaging with such suspended accounts, which enriches the dataset. Even further, the trolls' correlations get better controlled under such scenarios for the Russian and Chinese experiments.

Correlations Over Time

Among other things, Figure 4.4 shows that we can precisely predict troll activity regardless of restricting network sizes. On the other hand, unrestricted data experiments include time intervals where high activity peaks occur. To analyze our results over time, we directly observe how the diverse graph link set grows or shrinks in parallel to LP and NC scores. Figure 4.6 exposes these comparisons by adding a curve of *link sizes*, where every graph has been normalized to the largest one of each dataset within 5-day experiments.

Users		**Pearson's Correlations**		
Trolls	***Actives***	***Troll Sender Nodes***	***Active Sender Nodes***	***Receiver Nodes***
Russia	*closer*	-0.084	**0.110**	0.175
	distant	-0.083	0.430	0.090
	both	**0.007**	0.270	**0.044**
IRA	*closer*	**0.037**	0.120	-0.495
	distant	-0.450	0.260	-0.136
	both	-0.220	**0.030**	**-0.060**
China	*closer*	-0.400	0.077	**0.092**
	distant	0.078	0.160	0.327
	both	**0.007**	**0.005**	0.343

Table 4.5 – Given a whole run of experimental link prediction accuracy scores (over time), we compute the Pearson's correlation against the *three major network features*: number of troll nodes, number of active nodes, and the number of receiver nodes. We further divide each experiment setting according to trolls' place of origin and active users' distance from trolls.

Furthermore, we have only considered the settings where we mix both closer and distant active users.

The relationship between the red and blue curves in Figure 4.6 may also be understood, in general, under the *receiver column's* context, in Table 4.5. Nevertheless, while the link size curve depicts activity highs and lows, we highlight that our proposed pipeline can keep a *stable performance* at most times. Starting in August 2017, the IRA link prediction scores suffered from a lack of stability, thus, making it the only situation where abrupt changes in link sizes effectively correlate with the obtained results.

Despite having a high *positive correlation* with receiver nodes (as shown in Table 4.5), Figure 4.6 exhibits that Chinese link prediction scores do not get significantly decremented nor augmented as link sizes vary over time. In particular, within 2007, neither February-April nor July-September's size decrements nor September-November's increments alter the almost-constant link prediction accuracies.

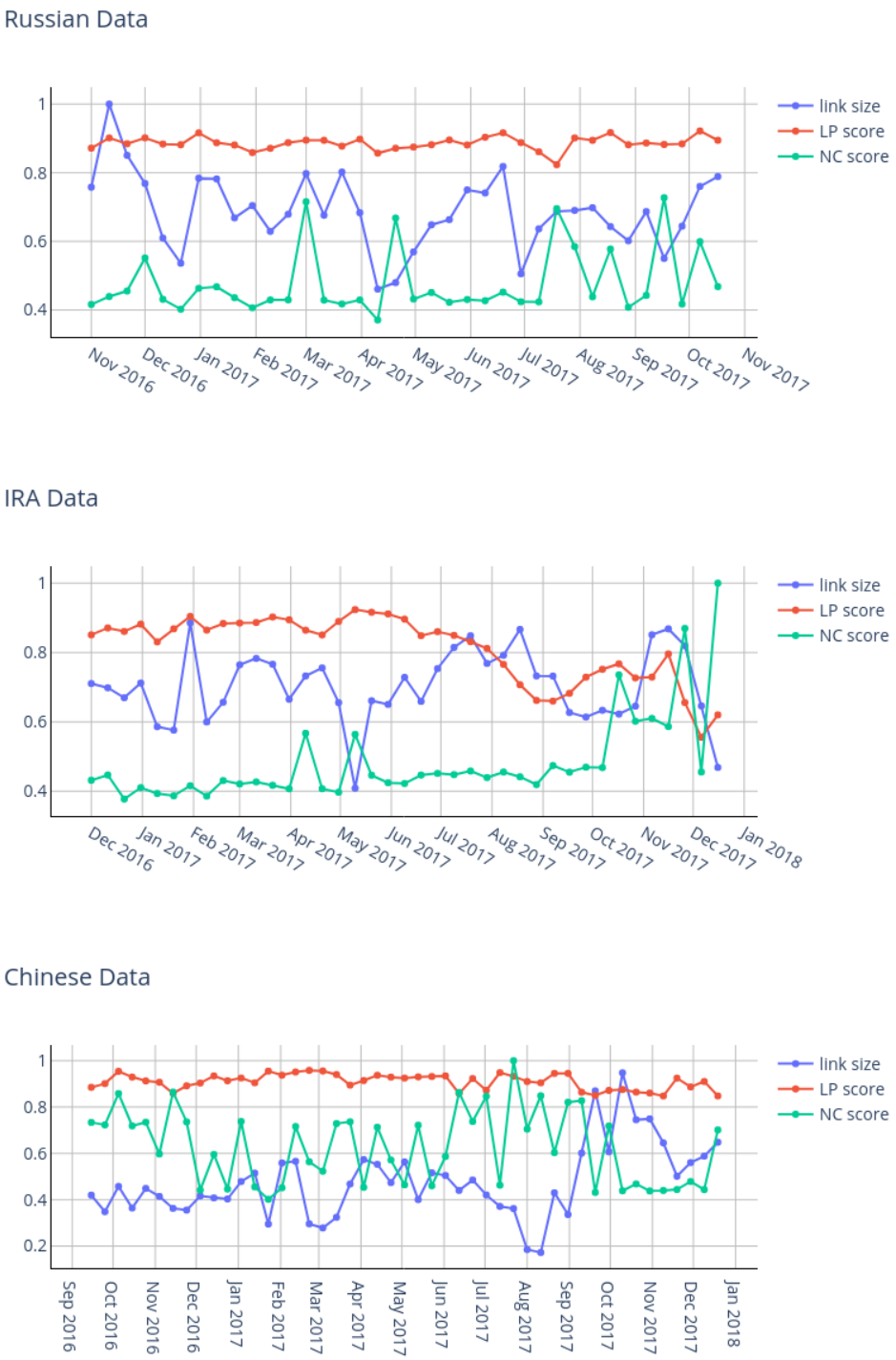


Figure 4.6 – We compare our F1 scores per time for the two tasks of interest in this project: link prediction (troll activity prediction) and node (troll) classification. We also add the relative size of the graph used for each corresponding experiment for all available activities. We highlight the stability achieved, in most cases, for link prediction, regardless of the number of available examples.

A Case Study: What Are Users Talking About?

For the following sections, we will spot some remarkable events which motivated the suspension of these Twitter accounts. This will help us to understand their characteristics as part of coordinated attacks on specific events. Moreover, given the good performances reported in Section 4.2, we consider showcasing our trained pipeline under an applied setting important. As any of our constructed networks are induced directly by tweets, we can derive a clustering procedure that is optimized from our LP loss yet agnostic to any text signal. Moreover, in Section 4.4, we will see that activity originating from the same type of users would tend to be clustered.

In December 2016, before Russia's 2018 election, anti-corruption blogger *Alexei Navalny* started his campaign. His movement style was unprecedented in modern-day Russia, as multiple sources compared it with anything in American politics. By focusing mainly on combating corruption and implementing macro-economic developments, Navalny constituted an opposition figure to President *Vladimir Putin* and Prime Minister *Dmitry Medvedev* (Harding, 2022; Nechepurenko, 2016). Within this context, the reported TEI (Russian) data release (Twitter, 2022), which includes actively promoting the *United Russia* While attacking the opposition, the party acquires its importance to understand how the tactics against Navalny's campaign were developed (Roth, 2020).

Motivated by the 2016 U.S. Presidential Election's role of social media, Twitter strengthened its strategies to detect coordinated accounts, especially those allegedly originating from the IRA. Hence, diverse strategies have been implemented to keep automated tabs out of Twitter. For our purposes, we considered November 2016, when the American Election took place. As multiple reports point out, the role of social media in amplifying *false claims* from *unreliable sources* highly made an impact on the election's outcome (Grinberg et al., 2019).

March 2017 was the month were several public debates took place en route to Hong Kong's Chief Executive Election (Haas, 2017). This month, the largest part of the campaigns was held until March 26th, the final election date. Media platforms constituted

an important part of this democratic process, as *pro-Beijing* interventions were claimed to be happening in favor of some candidates (Wood et al., 2019). Furthermore, as reported by Twitter's Information Operations team, these Chinese state-backed operations used diverse strategies to sow political discord in Hong Kong (Twitter, 2019).

We move forward with our analysis procedure on three significant months in our data: December 2016 for Russia, November 2016 for the IRA, and March 2017 for China. In Table 4.6, we present a set of *word clouds* which attempt to depict some of these trolls' main ideas during the selected months. We further detail how such topics relate to our LP embeddings in Section 4.4. The top words in the IRA datasets are the two leading candidates of the 2016 U.S. presidential elections. To understand these trolls better, we next study them more closely by looking at different topics of interest while contrasting them with active user topics.

Clustering Link Embeddings from Important Events

In Section 3.3.3, we introduced a method to reveal and cluster tweet topics from our learned link embeddings. To use such an idea in our context, we have taken one of *HuggingFace*'s implementation of the `RoBERTa` language model (Barbieri et al., 2020; Wolf et al., 2019), which was previously pretrained with around 58M tweets. This allows us to embed each tweet into a (high-dimensional) vector space. Furthermore, we have used `UMAP` (Campello et al., 2013) to compute equivalent embeddings on lower dimensions. The final vectors, $\tilde{H}_{\mathcal{M}}^{\top_G}$ can be analyzed even further with the help of clustering tools.

In Figure 4.7, we show three link embeddings from the specified dates of interest, from which we kick-start the tweet analysis mentioned before. The idea is, thus, to take any of the optimal clusters, depicted in Figure 4.7's lower row and recover the main topics in which users were engaged. In this case, it is worth recalling that a single cluster of links would bring together multiple social interactions. It is, therefore, not surprising to find diversity in each cluster's topics due to the language-agnostic nature of our DL pipeline.

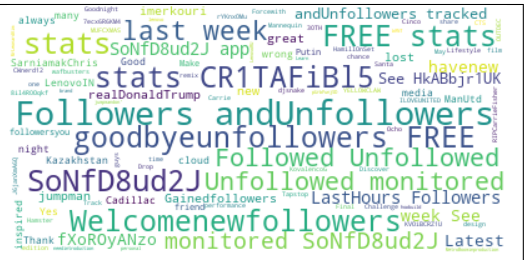
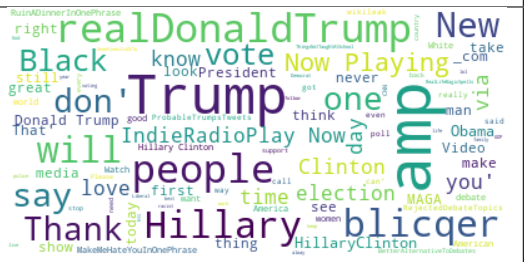
Dataset	*Dates*	*Main Event*	*Word Cloud of Troll's Tweets*
Russia	December 2016	Russian presidential elections	
IRA	November 2016	U.S. presidential election	
China	March 2017	Hong Kong Chief Executive election	

Table 4.6 – The event of interest studied around important elections in each dataset. The word clouds summarize the topics of tweets posted by trolls around the corresponding event.

In Tables 4.7, 4.8, and 4.9 we take three arbitrary clusters (from Figure 4.7) According to the described text embedding process, to discover some of the topics of interest that comprise these tweets. We divide these topics by user type, deriving into two columns that correspond to trolls or active users. We have not distinguished between closer and distant active users, as we did for past experiments.

We obtained multiple topics from the Russian dataset that do not necessarily fall into political discourse. For instance, in Table 4.7 (first row), we have depicted a cluster of trolls whose only intention is to announce their number of new *followers* from time to time; note that this process is often automatized and constitutes a common practice in

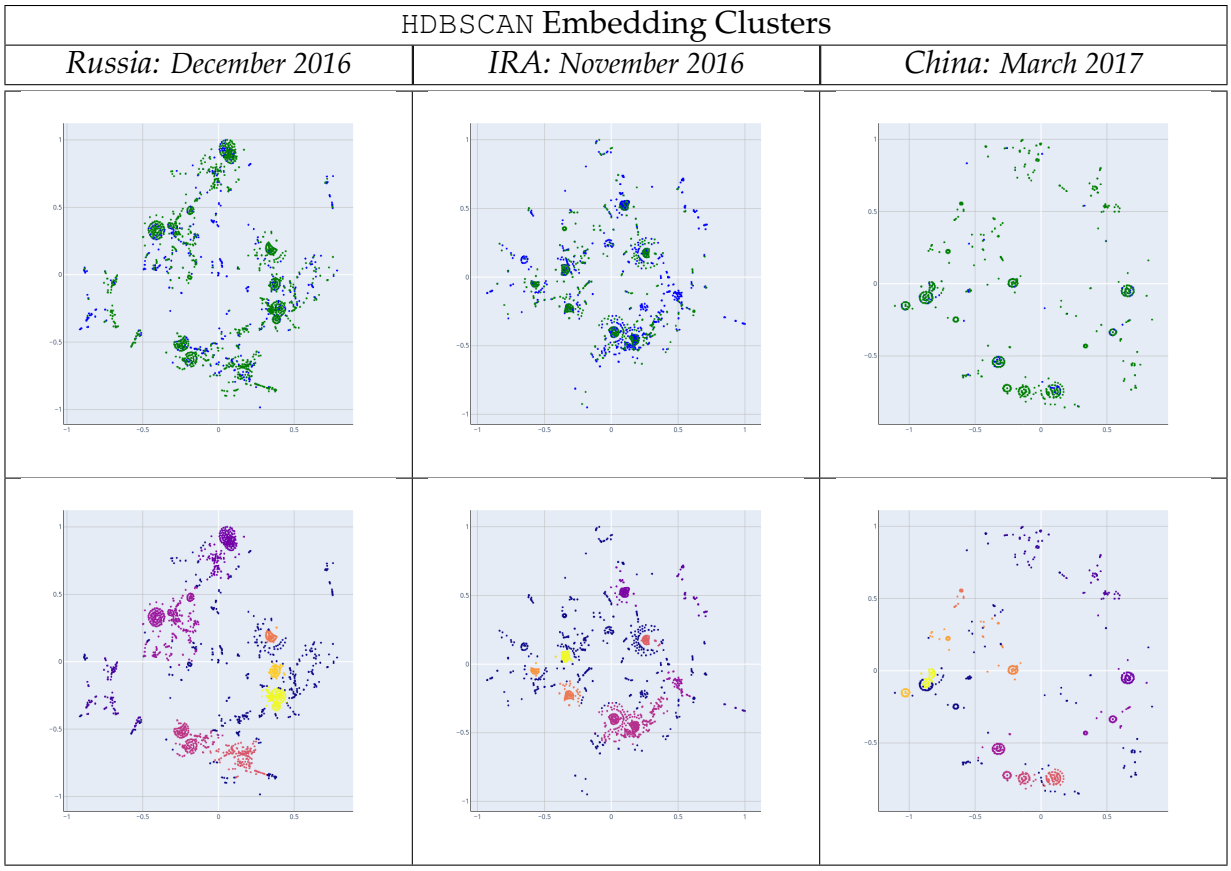


Figure 4.7 – Here we compare the classification labels obtained by our proposed LP pipeline (upper row) with the optimal clustering of embeddings (lower row) by `KMeans` (Jin and Han, 2010). Our model's ability to spatially group together Twitter activity can be assessed by two criteria: firstly, in the upper row, we could seek to find clusters of – mostly – exact colored embeddings; secondly, in the lower row, we could look at how similar to the posted tweets within each cluster.

Twitter. Moreover, our active user topics justify the diversity of their nature itself, as we can group topics like sports and Mexican politics in the same cluster, each other (Table 4.7, third row).

In the case of the explored IRA clusters, word clouds in Table 4.8 prioritize words and phrases related to the U.S. political atmosphere. This tendency prevails, even for active users, although topics do not necessarily concentrate on terms like `realDonaldTrump`, `MAGA`, or `Hillary Clinton`. The first and third rows also show several phrases in Spanish and Portuguese. Nonetheless, essential topics seem to keep around politics and news.

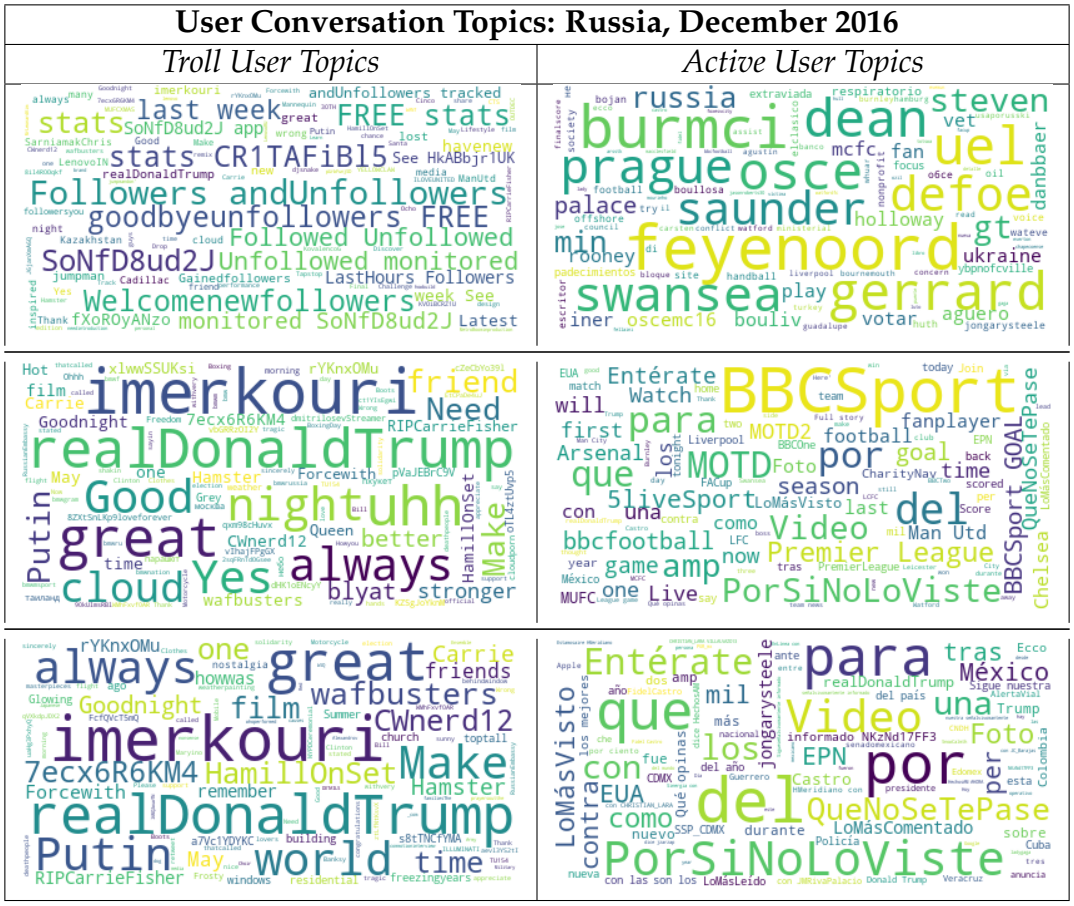


Table 4.7 – Within each row, two conversational word cloud cues are depicted, each classified according to its sender user's type. They both come from the same cluster (Figure 4.7) while diverging in the type of user producing each content. The first row contrasts a group of automated tweets programmed to display *Twitter following* statistics with users mainly talking about European soccer. In the second row, trolls mainly seem to talk about politics, while active ones prioritize European football. Finally, the third row roughly coincides with the trolls' second one, while active user tweets mostly come in the Spanish language; in particular, referring to Mexican news and politics.

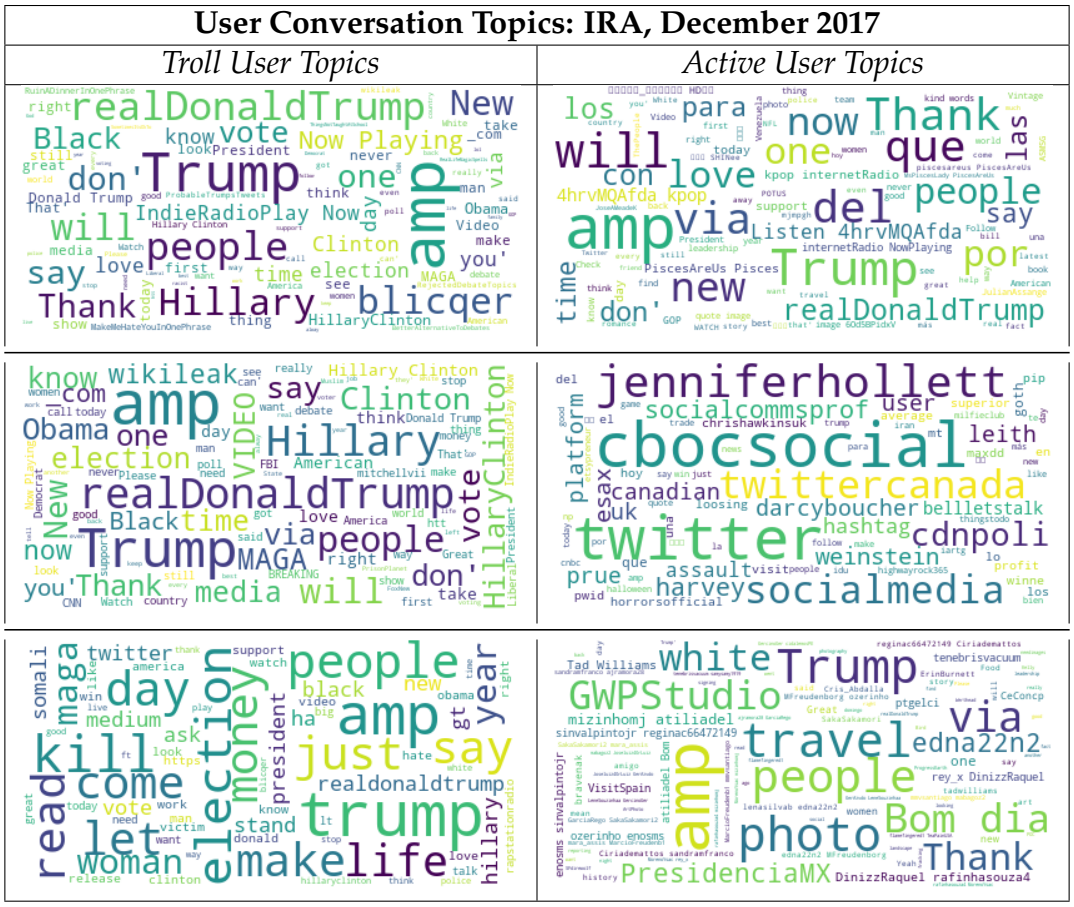


Table 4.8 – These word clouds were built following the same logic as in Table 4.7. Political topics predominate here, regardless of any user.

Finally, we highlight that we have masked any non-Latin alphabet characters for the Chinese topics. Recall that these trolls are targeting the political situation in Hong Kong. Nevertheless, we have been able to highlight some essential phrases from automated accounts (such as `ONLYRPE` or `senpaibot`). Active users, in this case, seem to be engaging in popular American political themes, yet interest in American sports also rises in the third row.

An important fact to highlight, within the context of the presented word clouds, relies on the lack of tweet text used to achieve the proposed clustering. Even if hashtag relations give information on topics of interest, we could show, in practice, that users with similar discourse also engage together in our studied social graphs. Moreover, from Tables 4.7,

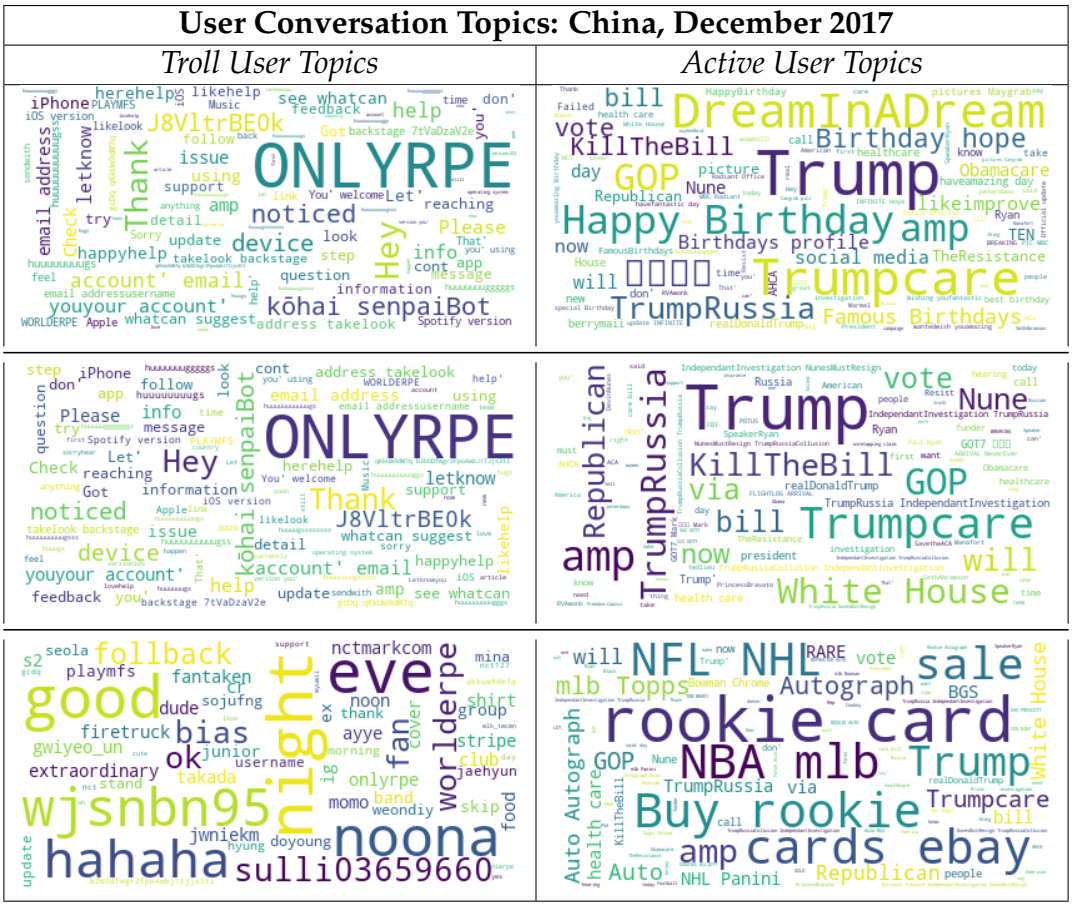


Table 4.9 – Once again, we follow the same logic for these word clouds as presented in Tables 4.8 and 4.7. Despite this dataset's origin, no Chinese characters are present here, as we focus on directly interpreting what users say. In this case, active users were the ones who talked about worldwide known U.S. politics and sports themes.

4.8, and 4.9, we see that active users have a wider variety of topics of interest: this is not surprising as the released set of trolls is, by definition, meant to be part of coordinated attacks to concrete goals. Nonetheless, our contribution allows us to better understand massive language communication patterns in social media by mainly focusing on user-to-user interactions.

Chapter 5

Conclusion

In this thesis, we have performed extensive experiments to investigate confirmed troll activities over a long time in three different datasets, using recent graph representation learning techniques. In particular, we show that learning an encoding of user activities, over time, allows studying trolls. Below we summarize our work.

5.1 Summary of Work

The present work attempts to capture Twitter's account suspension criteria for massively coordinated accounts. Thus, we consider the availability of this data to raise the critical question of investigating whether these trolls' online behavior would have anything to do with that from *active users*. In this sense, we acknowledge the existence of a wide variety of communities within Twitter; hence have decided to focus only on the direct interactions produced by released accounts (see Section 3.1).

We collected three troll account datasets, each one originating in *Russia*, *China*, and Russian's *Internet Research Agency* (IRA). These timestamped data comprise a list of tweets accompanied by metadata, tweet text, and mentioned users and hashtags. All was made public by Twitter's Information Operations efforts, as detailed in Section 3.1.1. Furthermore, we used the available hashtags and user mentions to acquire sets of non-troll users

as control samples for each troll dataset. Moreover, we have made sure these users remain *active* in Twitter to date, that is, they have not been suspended. Section 3.2.2 details how we combined both datasets with building mention-hashtag graphs. These networks connect *three types* of nodes, according to actual Twitter activity: **trolls** and **non-trolls** mentioning **hashtags**, or any other user.

The controlled set sampling is explained in Figure 4.1; it is similar to what is expected in the literature for sampling a random set of Twitter users (i.e., using a seed set of keywords), as explained in Section 2.7. We refer to this control set as **active** users since we only know, for sure, that they were non-suspended around the same time as trolls (going back to 2016) and remained active to date. More specifically, we have distinguished a closer from a distant set of active users, depending on their distance from the original collection of trolls.

Moreover, in Section 3.3.1, we went through our feature extraction process, in which we defined a set of metapaths according to each node type. These abstractions helped us compute necessary node features for our link prediction pipeline. The pipeline has been described in Section 3.3.2: we learned an *activity* (link) prediction model by feeding the node as mentioned earlier features, by training on the DGCNN model (as described in Section 2.3.2). This training process provides a measure of predictability for *user to user* and *user to hashtag* distributions. At the same time, we have described methods to train *user* (node) classification and *unsupervised* clustering from the learned deep embeddings (Section 3.3.3).

To implement our approach, we have designed an experimental pipeline to extract activity from predefined granularities (time intervals), namely 5, 10, 30, or 60 days. Within a window of time, in Section 4.1, we detail how we obtain networks of troll versus active user interactions, as well as data balancing concerns (Table 4.2). Our *activity prediction* results are presented in Section 4.2, in which we assess macro-F1 scores per time interval. We have also shown an ablation study in which we skip the node labeling mechanism proposed by Zhang and Chen (2018).

Our link prediction assessment, then, moves forward to node classification, as described in Section 4.3. We have presented F1 scores and time interval comparisons of how these two tasks perform – and vary – concerning data availability. With this in mind, Section 4.4 discusses how dataset size would affect our results regarding feature correlations. Moreover, we reported and compared the results obtained for both balanced and imbalanced settings, as depicted in Figures B.2, B.3, and B.4. We observed similar patterns, but the variance is lower in the balanced setting. Table 4.1 shows that most nodes in our constructed graphs are mentioned users and hashtags. The balancing we are applying mainly affects these two types of nodes and senders. We explained this process in Section 3.2.2.

To dig deeper into what the proposed pipeline was learning, we studied the effect of node labeling (a key component of DGCNN) in our task, as reported in Table 4.3. Table B.2 provides further results, which confirm that this helps with performance. We also studied whether the performance of the method used is correlated with certain data features in a particular size of data (which can be confounding factors). We did not have these in the initial submission but included them in this version. In particular, we use Pearson's correlations to quantify performance correlation with node sizes of different types in Table 4.5, while Figure 4.6 shows performance correlates with the number of links over time. We observed that overall link size is not a key factor as it stays in the same range. Some significant correlations should be considered for the node sizes when interpreting the results.

Finally, in Section 4.4 we have utilized three particular time intervals, that corresponded to three important for each of the Russian, IRA, and Chinese datasets. In particular, we zoomed into activities around three major Elections in our three datasets. We showed that the clustering procedure explained in Section 3.3.3 can help identify users communicating on related topics, even though our method did not use any text signal. Tables 4.6, 4.7, 4.8, 4.9, as well as, Figure 4.7 further expand our findings.

5.2 Contributions

Our pipeline started with building heterogeneous graphs via trolls and non-trolls interactions on Twitter. The presented data curation process has value in that while direct and indirect *user interactions* are encoded by links, information about what users are talking about is preserved without including text features. This fact comes in handy in the ways our model can learn user activity; yet, firstly, we shall emphasize that our data categorization, given *three places of origin*, brings up a way to analyze the TEI dataset, which is unusual in previous work (Sharma et al., 2021; Zannettou et al., 2019a,c).

Furthermore, we have used *active* Twitter users to characterize non-trolls. This has allowed us to use any of the given troll datasets to crawl corresponding control sets. Moreover, our data acquisition pipeline (Section 4.3) can work with multiple relationships. In particular, non-troll data augmentation started from *hashtag relations* for the Russian and Chinese datasets. On the other hand, the IRA one must take an extra crawling step, starting from *user mentions*, due to the lack of available hashtags.

Our link prediction pipeline can achieve 83.75%, 76.25%, and 87.75% average F1 validation scores for the Russian, IRA, and Chinese datasets, respectively. These results have been broken down into multiple experiments, whose settings depend mainly on data sampling time intervals (Table 4.2). Due to the lack of significant related work, we have opted to verify the proposed model's node tagging component to argue its importance for activity prediction. And indeed, Table 4.3 shows that doing such ablation decreases performance overall. Furthermore, Figure B.2 shows that link prediction accuracy for active users is higher than for troll users; however, the accuracy gap is more significant for the distant users compared to closer ones.

Moreover, we have shown that for each type of user activity, the distribution of correctly predicted links mostly lies above 80% for all datasets. The Chinese dataset has its troll user activity *correctly classified* more than its non-trolls, which is not what happens for the other two datasets (Figure 4.4). With this notion, we could quantify how predictable

users' behaviors in each dataset are. In this sense, Russian and IRA active users are *more predictable* than trolls. Our results also, not surprisingly, reflected the complexity of the IRA dataset, as it was the most dispersed in terms of per-class accuracy proportions; this outcome can be explained by its lack of hashtag nodes.

Regarding node classification, Table 4.4 showed low F1 scores, almost close to 50% for most settings. Yet notably, the Chinese dataset reached 87% F1 score in experiments sampling graphs at 5-day intervals. This number, nevertheless, exceeds the best F1 score reported in related work by Sharma et al. (2021), in which results are only reported for the IRA dataset. Once again, this work's outcome would contribute to contrasting different troll datasets, which form part of the same social network while designed for different coordinated attack purposes. Furthermore, this thesis shows that the patterns observed in other datasets are different.

One significant advantage of our approach arises from the lack of dependency between graph features and link prediction results. We measured this in terms of correlations concerning the number of senders (trolls or active users) and receiver nodes (proportional to the total number of links). Table 4.5 shows that experiments that included *distant* (2-hop) active users tend to lower their correlations; and this happens for each of the three datasets and at least one of the graphs mentioned above features. Moreover, our LP scores over time were not affected by the availability of data (Figure 4.6).

The main takeaway of this thesis is that learning and encoding user activities over a fixed time contributes to learning certain troll features that would help classify them from non-troll users. These encodings are learned from the user activities when the learning objective is to predict them. This learning ignores the user type; troll and active users are combined and treated the same.

To conclude our result interpretations, we used some of our learned link embeddings, resulting from the implemented *DGCNN* network, to show clusters of user activity and to further discover how the model was able to sub-classify links by additional features (Figure 4.7). We were able to compute optimal clusters; per each point in space, a tweet

was sent, which means some text phrase was posted on Twitter. Hence, we extracted the available text per such points and used a language model (Barbieri et al., 2020) to compute their most important topics. We observed three important events per dataset to assess our findings, mainly related to electoral processes.

In this thesis, we have also investigated if the learned user embeddings can be used for troll detection (Section 4.3) and studied their engagement (Section 4.4). We showed that while automatic detection is not achievable (the classification accuracy is close to random in two of the three datasets, and only the third less sophisticated trolls in China dataset are detectable: see Table 4.6); these embeddings are useful for investigating the activities of trolls and their engagement with the other users. In particular, we can compare the discussion topics of trolls and active users using these encodings when zooming in around a specific date. For example, in Table 4.8, we have shown the main topics around the 2016 US Election while contrasting them with the engagement of the IRA trolls in each topic with that of the control group.

Our embeddings could then semantically cluster tweets together while agnostic to any text feature. For instance, we obtained a Spanish language cluster and some sports-related sports-related clusters from the Russian dataset (Table 4.7). A commonality for all datasets is that we could always find clusters talking about American politics; yet, this coincides only with Russian and IRA trolls, while Chinese active users were the counterpart who was. Interested in such a topic.

5.3 Future Work

Many essential and relevant features available within the TEI dataset have remained unexplored for the scope of this thesis. Thus, future work is crucial to enable new ways of understanding coordinated accounts. For starters, how we have used the provided tweet timestamps is irrelevant at neural training time, as our mentioned graphs do not acquire

any abstraction from them. Hence, it is important to leverage these datasets' *temporal* nature, which affects their structural properties at different intervals.

Another important, and yet popular, area of opportunity relies on the realm of learning *causal factors* that directly affect the occurrence of large-scale events. For instance, sets of coordinated accounts together usually perform the same action to give relevance to specific content, such as retweeting misinformed posts, massively following other particular sources of fake news, or immediately *canceling out* diverse opinions about a political event. We believe this behavior could be modeled as a function of "harmless events" such as knowing *a priori* that a novel user is staying within its niche of accounts or is "*attacking*" specific controversial hashtags.

The release of these state-backed accounts also comes with an extensive data collection of media files that have been crucial for their purposes. Previous work and reports have stressed the importance of visual details towards the process of *viralizing* ideas, regardless of their trustworthiness, within different Twitter communities. Evidence has been brought on adopting these patterns, in the form of *Internet memes*, by various groups of users to transmit their ideas. Hence, work shall be done to incorporate the transmission and evolution of the provided image memes as an essential vehicle for misinformation spreading.

5.4 Concluding Remarks

The Internet has become an essential part of daily life, and during the COVID-19 pandemic, social media has been a vital way for people to stay connected. However, this ease of communication also makes it possible for organizations to spread false information to influence public opinion. The rapid flow of data on the Internet does not ensure truthfulness, which specific organizations have exploited to develop strategies of massive opinion manipulation to shape the outcome of events in a predetermined manner.

As machine learning practitioners, we believe providing the community with tools to help uncover the true objectives of large-scale online events is essential. We view our work as compared to what social science seeks to understand about human relationships, but it also leverages the rapidly growing field of graph representation learning. We believe that discussions on best practices for online behavior and education in t his area are necessary. We aim to empower the community with tools to unveil the true intentions behind massive online events.

Finally, we recognize and embrace the tremendous technological advancements that have kept people connected. We actively work on making this environment as safe and friendly as the external world could be. And more importantly, in an openly designed space like Twitter, we acknowledge, promote, and take action towards *defending every user's freedom of speech*.

Bibliography

Information operations, 2021. URL https://transparency.twitter.com/en/reports/information-operations.html.

L. A. Adamic and E. Adar. Friends and neighbors on the web. *Soc. Networks*, 25:211–230, 2003.

A. Addawood, A. Badawy, K. Lerman, and E. Ferrara. Linguistic cues to deception: Identifying political trolls on social media. *Proceedings of the International AAAI Conference on Web and Social Media*, 13(01):15–25, Jul. 2019. URL https://ojs.aaai.org/index.php/ICWSM/article/view/3205.

A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 37–48, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320351. doi: 10.1145/2488388.2488393. URL https://doi.org/10.1145/2488388.2488393.

D. Alba. How russia's troll farm is changing tactics before the fall election, 2020.

A. Badawy, A. Addawood, K. Lerman, and E. Ferrara. Characterizing the 2016 russian IRA influence campaign. *CoRR*, abs/1812.01997, 2018a. URL http://arxiv.org/abs/1812.01997.

A. Badawy, K. Lerman, and E. Ferrara. Who falls for online political manipulation? *CoRR*, abs/1808.03281, 2018b. URL http://arxiv.org/abs/1808.03281.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

A. V. Banerjee. A Simple Model of Herd Behavior*. *The Quarterly Journal of Economics*, 107(3):797–817, 08 1992. ISSN 0033-5533. doi: 10.2307/2118364. URL https://doi.org/10.2307/2118364.

A.-L. Barabási and M. Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016. ISBN 9781107076266 1107076269. URL http://barabasi.com/networksciencebook/.

F. Barbieri, J. Camacho-Collados, L. Espinosa Anke, and L. Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.148. URL https://aclanthology.org/2020.findings-emnlp.148.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, page 585–591, Cambridge, MA, USA, 2001. MIT Press.

S. Bikhchandani, D. Hirshleifer, and I. Welch. A theory of fads, fashion, custom, and cultural change as informational cascades. *Journal of Political Economy*, 100(5):992–1026, 1992. ISSN 00223808, 1537534X. URL http://www.jstor.org/stable/2138632.

K. Browne. Snowball sampling: using social networks to research non-heterosexual women. *International Journal of Social Research Methodology*, 8(1):47–60, 2005. doi: 10.1080/1364557032000081663. URL https://doi.org/10.1080/1364557032000081663.

R. J. G. B. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37456-2.

S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 891–900, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337946. doi: 10.1145/2806416.2806512. URL https://doi.org/10.1145/2806416.2806512.

G. L. Ciampaglia, P. Shiralkar, L. M. Rocha, J. Bollen, F. Menczer, and A. Flammini. Computational fact checking from knowledge networks. *CoRR*, abs/1501.03471, 2015. URL http://arxiv.org/abs/1501.03471.

S. Dixon. Global social networks ranked by number of users 2022, Jul 2022. URL https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/.

H. Dorney. How to create and use hashtags, 2021. URL https://business.twitter.com/en/blog/how-to-create-and-use-hashtags.html.

D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, USA, 2010. ISBN 0521195330.

S. Evanega, M. Lynas, J. Adams, and K. Smolenyak. Coronavirus misinformation: quantifying sources and themes in the covid-19 'infodemic' (preprint). 10 2020. doi: 10.2196/preprints.25143.

E. Ferrara. What types of covid-19 conspiracies are populated by twitter bots? *First Monday*, May 2020. ISSN 1396-0466. doi: 10.5210/fm.v25i6.10633. URL http://dx.doi.org/10.5210/fm.v25i6.10633.

V. Gadde and Y. Roth. Enabling further research of information operations on twitter, 2018a. URL https://blog.twitter.com/en_us/topics/company/2018/enabling-further-research-of-information-operations-on-twitter.

V. Gadde and Y. Roth. Enabling further research of information operations on twitter, Oct 2018b. URL https://blog.twitter.com/en_us/topics/company/2018/enabling-further-research-of-information-operations-on-twitter.

R. Gallotti, F. Valle, N. Castaldo, P. Sacco, and M. D. Domenico. Assessing the risks of 'infodemics' in response to COVID-19 epidemics. *Nature Human Behaviour*, 4(12):1285–1293, oct 2020. doi: 10.1038/s41562-020-00994-6. URL https://doi.org/10.1038%2Fs41562-020-00994-6.

M. Gjoka, C. T. Butts, M. Kurant, and A. Markopoulou. Multigraph sampling of online social networks. *IEEE Journal on Selected Areas in Communications*, 29(9):1893–1905, 2011. doi: 10.1109/JSAC.2011.111012.

N. Grinberg, K. Joseph, L. Friedland, B. Swire-Thompson, and D. Lazer. Fake news on twitter during the 2016 u.s. presidential election. *Science*, 363(6425):374–378, 2019. doi: 10.1126/science.aau2706. URL https://www.science.org/doi/abs/10.1126/science.aau2706.

A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016. URL http://arxiv.org/abs/1607.00653.

B. Haas. Hong kong elects a new chief executive: What you need to know, Mar 2017. URL https://www.theguardian.com/world/2017/mar/22/hong-kong-chief-executive-election-what-you-need-to-know.

W. L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. URL http://arxiv.org/abs/1706.02216.

K. Hao. Troll farms reached 140 million americans a month on facebook before 2020 election, internal report shows, Sep 2021. URL https://www.technologyreview.com/2021/09/16/1035851/facebook-troll-farms-report-us-2020-election/.

L. Harding. Alexei navalny calls for social media 'information front' against russia, Apr 2022. URL https://www.theguardian.com/world/2022/apr/14/alexei-navalny-calls-for-social-media-information-front-against-russia.

J. Harmon. The Psychology of Interpersonal Relations. By Fritz Heider. New York: John Wiley and Sons, Inc., 1958. 322 pp. 6.25. *Social Forces*, 37(3):272–273, 03 1959. ISSN 0037-7732. doi: 10.2307/2572978. URL https://doi.org/10.2307/2572978.

L. Havrlant and V. Kreinovich. A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation). *International Journal of General Systems*, 46:27–36, 01 2017. doi: 10.1080/03081079.2017.1291635.

P.-M. Hui, C. Shao, A. Flammini, F. Menczer, and G. L. Ciampaglia. The hoaxy misinformation and fact-checking diffusion network, 2018. URL https://aaai.org/ocs/index.php/ICWSM/ICWSM18/paper/view/17851.

F. Jin, W. Wang, L. Zhao, E. Dougherty, Y. Cao, C. Lu, and N. Ramakrishnan. Misinformation propagation in the age of twitter. *Computer*, 47(12):90–94, dec 2014. ISSN 1558-0814. doi: 10.1109/MC.2014.361.

X. Jin and J. Han. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_425. URL https://doi.org/10.1007/978-0-387-30164-8_425.

J. Kao, R. Zhong, P. Mozur, and A. Krolik. How china spreads its propaganda version of life for uyghurs, Jun 2021. URL https://www.propublica.org/article/how-china-uses-youtube-and-twitter-to-spread-its-propaganda-version-of-life-for-uyghurs-in-xinjiang.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

D. Klepper. Cyborgs, trolls, and bots: A guide to online misinformation, 2020. URL https://apnews.com/article/us-news-ap-top-news-elections-social-media-technology-4086949d878336f8ea6daa4dee725d94.

M. Kurant, M. Gjoka, Y. Wang, Z. W. Almquist, C. T. Butts, and A. Markopoulou. Coarse-grained topology estimation via graph sampling. *CoRR*, abs/1105.5488, 2011a. URL http://arxiv.org/abs/1105.5488.

M. Kurant, A. Markopoulou, and P. Thiran. Towards unbiased bfs sampling. *IEEE Journal on Selected Areas in Communications*, 29:1799–1809, 2011b.

P. J. Laub, T. Taimre, and P. K. Pollett. Hawkes processes, 2015.

J. Leskovec and C. Faloutsos. Sampling from large graphs. 2006.

D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, page 556–559, New York, NY, USA, 2003a. Association for Computing Machinery. ISBN 1581137230. doi: 10.1145/956863.956972. URL https://doi.org/10.1145/956863.956972.

D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, 2003b.

X. Lou, A. Flammini, and F. Menczer. Information pollution by social bots. *CoRR*, abs/1907.06130, 2019. URL http://arxiv.org/abs/1907.06130.

L. Luceri, S. Giordano, and E. Ferrara. Don't feed the troll: Detecting troll behavior via inverse reinforcement learning. *CoRR*, abs/2001.10570, 2020. URL https://arxiv.org/abs/2001.10570.

J. Ma, W. Gao, and K.-F. Wong. Rumor detection on Twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1184. URL https://www.aclweb.org/anthology/P18-1184.

L. McInnes and J. Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *ArXiv*, abs/1802.03426, 2018.

L. McInnes, J. Healy, and S. Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar 2017. doi: 10.21105/joss.00205. URL https://doi.org/10.21105%2Fjoss.00205.

S. A. Memon and K. M. Carley. Characterizing COVID-19 misinformation communities using a novel twitter dataset. *CoRR*, abs/2008.00791, 2020. URL https://arxiv.org/abs/2008.00791.

M. Mendoza, M. Tesconi, and S. Cresci. Bots in social and interaction networks: Detection and impact estimation. *ACM Trans. Inf. Syst.*, 39(1), Oct. 2020. ISSN 1046-8188. doi: 10.1145/3419369. URL https://doi.org/10.1145/3419369.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

R. L. Murphy, B. Srinivasan, V. A. Rao, and B. Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *CoRR*, abs/1811.01900, 2018. URL http://arxiv.org/abs/1811.01900.

I. Nechepurenko. Aleksei navalny, putin critic, says he'll run for president of russia, Dec 2016. URL https://www.nytimes.com/2016/12/13/world/europe/aleksei-navalny-putin-president-russia.html.

M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45 (2):167–256, Jan 2003. ISSN 1095-7200. doi: 10.1137/s003614450342480. URL http://dx.doi.org/10.1137/S003614450342480.

M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1105–1114, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939751. URL https://doi.org/10.1145/2939672.2939751.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999a. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999b. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120.

G. Pennycook, Z. Epstein, M. Mosleh, A. Arechar, D. Eckles, and D. Rand. Understanding and reducing the spread of misinformation online. *ACR North American Advances*, 2020.

J. E. Ramos. Using tf-idf to determine word relevance in document queries. 2003.

A. Rapoport. Spread of information through a population with socio-structural bias: I. assumption of transitivity. *The bulletin of mathematical biophysics*, 15(4):523–533, 1953. doi: 10.1007/BF02476440. URL https://doi.org/10.1007/BF02476440.

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

Y. Roth. Disclosing networks of state-linked information operations we've removed, Jun 2020. URL https://blog.twitter.com/en_us/topics/company/2020/information-operations-june-2020.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, Dec. 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009. ISBN 0136042597.

D. Ruths. The misinformation machine. *Science*, 363(6425):348–348, 2019. doi: 10.1126/science.aaw1315.

D. Ruths and J. Pfeffer. Social media for large studies of behavior. *Science*, 346(6213): 1063–1064, 2014. doi: 10.1126/science.346.6213.1063.

K. Sharma, S. Seo, C. Meng, S. Rambhatla, A. Dua, and Y. Liu. Coronavirus on social media: Analyzing misinformation in twitter conversations. *CoRR*, abs/2003.12309, 2020. URL https://arxiv.org/abs/2003.12309.

K. Sharma, Y. Zhang, E. Ferrara, and Y. Liu. Identifying coordinated accounts on social media through hidden influence and group behaviours. In *Proceedings of the 27th*

ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '21, page 1441–1451, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467391. URL https://doi.org/10.1145/3447548.3467391.

K. Shu, D. Mahudeswaran, S. Wang, and H. Liu. Hierarchical propagation networks for fake news detection: Investigation and exploitation. *CoRR*, abs/1903.09196, 2019a. URL http://arxiv.org/abs/1903.09196.

K. Shu, S. Wang, and H. Liu. Beyond news contents: The role of social context for fake news detection. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 312–320, New York, NY, USA, 2019b. Association for Computing Machinery. ISBN 9781450359405. doi: 10.1145/3289600.3290994. URL https://doi.org/10.1145/3289600.3290994.

L. G. Stewart, A. Arif, and K. Starbird. Examining trolls and polarization with a retweet network. In *Proc. ACM WSDM, Workshop on Misinformation and Misbehavior Mining on the Web.*, 2018. URL https://faculty.washington.edu/kstarbi/examining-trolls-polarization.pdf.

Twitter. Information operations directed at hong kong, Aug 2019. URL https://blog.twitter.com/en_us/topics/company/2019/information_operations_directed_at_Hong_Kong.

Twitter. Twitter information operations, 2022. URL https://transparency.twitter.com/en/reports/information-operations.html.

L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax, 2018.

O. Vinyals, S. Bengio, and M. Kudlur. Order matters: Sequence to sequence for sets, 2016.

D. Volchek. Inside the 'propaganda kitchen' – a former russian 'troll factory' employee speaks out, 2021. URL https://www.rferl.org/a/russian-troll-factory-hacking/31076160.html.

T. Wang, Y. Chen, Z. Zhang, T. Xu, L. Jin, P. Hui, B. Deng, and X. Li. Understanding graph sampling algorithms for social network analysis. In *2011 31st international conference on distributed computing systems workshops*, pages 123–128. IEEE, 2011.

Y. Wang, M. McKee, A. Torbica, and D. Stuckler. Systematic literature review on the spread of health-related misinformation on social media. *Social Science Medicine*, 240:112552, 2019. ISSN 0277-9536. doi: https://doi.org/10.1016/j.socscimed.2019.112552. URL https://www.sciencedirect.com/science/article/pii/S0277953619305465.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019. URL http://arxiv.org/abs/1910.03771.

J. C. Wong. Russian agency created fake leftwing news outlet with fictional editors, facebook says, 2020. URL https://www.theguardian.com/technology/2020/sep/01/facebook-russia-internet-research-agency-fake-news.

D. Wood, S. McMinn, and E. Feng. China used twitter to disrupt hong kong protests, but efforts began years earlier, Sep 2019. URL https://www.npr.org/2019/09/17/758146019/china-used-twitter-to-disrupt-hong-kong-protests-but-efforts-began-years-earlier.

K. Yang, P. Hui, and F. Menczer. Bot electioneering volume: Visualizing social bot activity during elections. *CoRR*, abs/1902.02339, 2019. URL http://arxiv.org/abs/1902.02339.

M. I. Yousuf and S. Kim. Guided sampling for large graphs. *Data Mining and Knowledge Discovery*, 34:905–948, 2020.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. Deep sets. *CoRR*, abs/1703.06114, 2017. URL http://arxiv.org/abs/1703.06114.

S. Zannettou, T. Caulfield, J. Blackburn, E. D. Cristofaro, M. Sirivianos, G. Stringhini, and G. Suarez-Tangil. On the origins of memes by means of fringe web communities. *CoRR*, abs/1805.12512, 2018. URL http://arxiv.org/abs/1805.12512.

S. Zannettou, B. Bradlyn, E. D. Cristofaro, G. Stringhini, and J. Blackburn. Characterizing the use of images by state-sponsored troll accounts on twitter. *ArXiv*, abs/1901.05997, 2019a.

S. Zannettou, T. Caulfield, E. De Cristofaro, M. Sirivianos, G. Stringhini, and J. Blackburn. Disinformation warfare: Understanding state-sponsored trolls on twitter and their influence on the web. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 218–226, New York, NY, USA, 2019b. Association for Computing Machinery. ISBN 9781450366755. doi: 10.1145/3308560.3316495. URL https://doi.org/10.1145/3308560.3316495.

S. Zannettou, T. Caulfield, W. Setzer, M. Sirivianos, G. Stringhini, and J. Blackburn. Who let the trolls out? towards understanding state-sponsored trolls. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '19, page 353–362, New York, NY, USA, 2019c. Association for Computing Machinery. ISBN 9781450362023. doi: 10.1145/3292522.3326016. URL https://doi.org/10.1145/3292522.3326016.

M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018.

M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

Y. Zhang, E. D. Kolaczyk, and B. D. Spencer. Estimating network degree distributions under sampling: An inverse problem, with applications to monitoring social media networks. *The Annals of Applied Statistics*, 9:166–199, 2015.

X. Zhou and R. Zafarani. Network-based fake news detection: A pattern-driven approach. *CoRR*, abs/1906.04210, 2019. URL http://arxiv.org/abs/1906.04210.

X. Zhou and R. Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Comput. Surv.*, 53(5), Sept. 2020. ISSN 0360-0300. doi: 10.1145/3395046. URL https://doi.org/10.1145/3395046.

Appendix A

Further Dataset Descriptions

A.1 Monthly Statistics

In this section, we attempt to compare the distribution of obtained data varying with time. Given the available Russian troll and active user data, Figure A.1 displays each data source's contribution while also giving the total amount of tweets per month on the y-axis. Figures A.2 and A.3 present analogous plots for the IRA and Chinese datasets, respectively. In any case, active users dominate the number of trolls per month; furthermore, the total percentage of users can be found in Table 3.1.

A.2 All Time Troll Activity

Figure 4.2 provides information on any activity peaks by trolls, closer, and distant active users. Due to data availability restrictions, those numbers of tweets were truncated to guarantee the existence of links for any experiment from any data source. Nonetheless, it is worth noting that suspension trolls often occurred way before the considered dates (such as the ones we have studied in Section 4.4). Hence, we depict all-time available troll activity in Figure A.4. For our three datasets, it was possible to go even before 2010; yet it is evident that activity considerably increases as major political events come by.

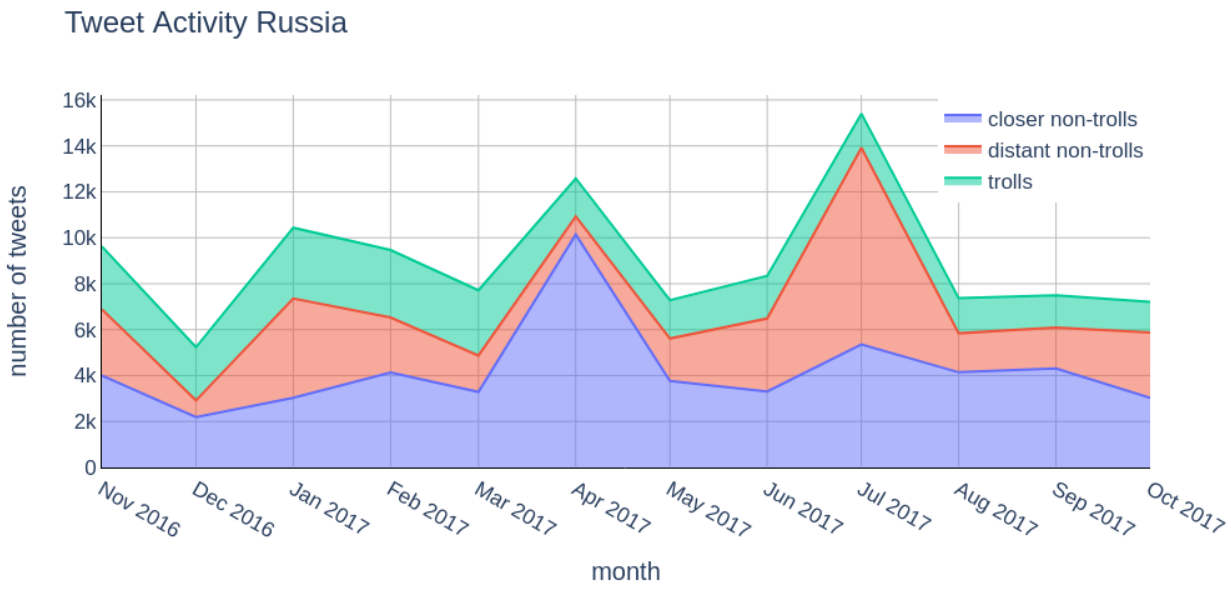


Figure A.1 – The number of tweets per month for the Russian dataset. Troll activity is compared with the one generated by closer and distant active users.

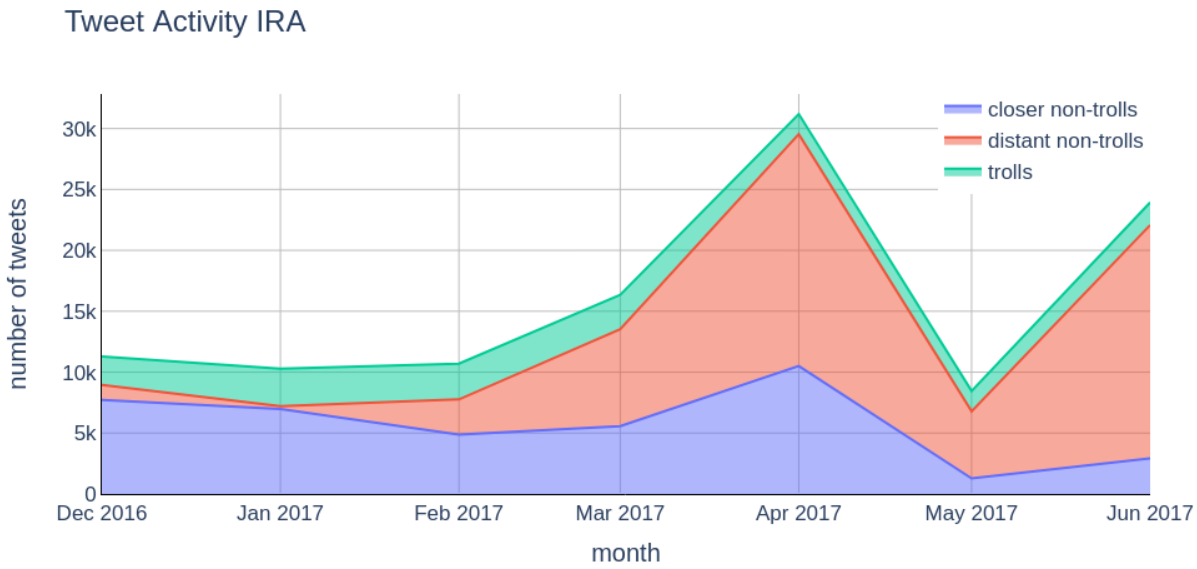


Figure A.2 – The number of tweets per month for the IRA dataset. Troll activity is compared with the one generated by closer and distant active users.

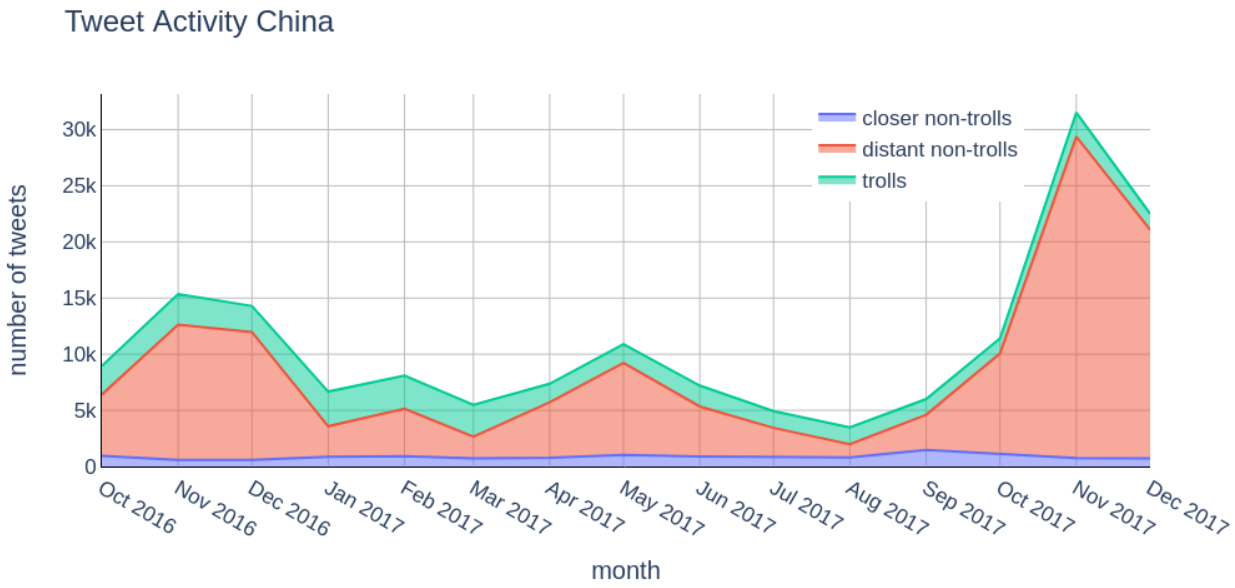


Figure A.3 – The number of tweets per month for the Chinese dataset. Troll activity is compared with the one generated by closer and distant active users.

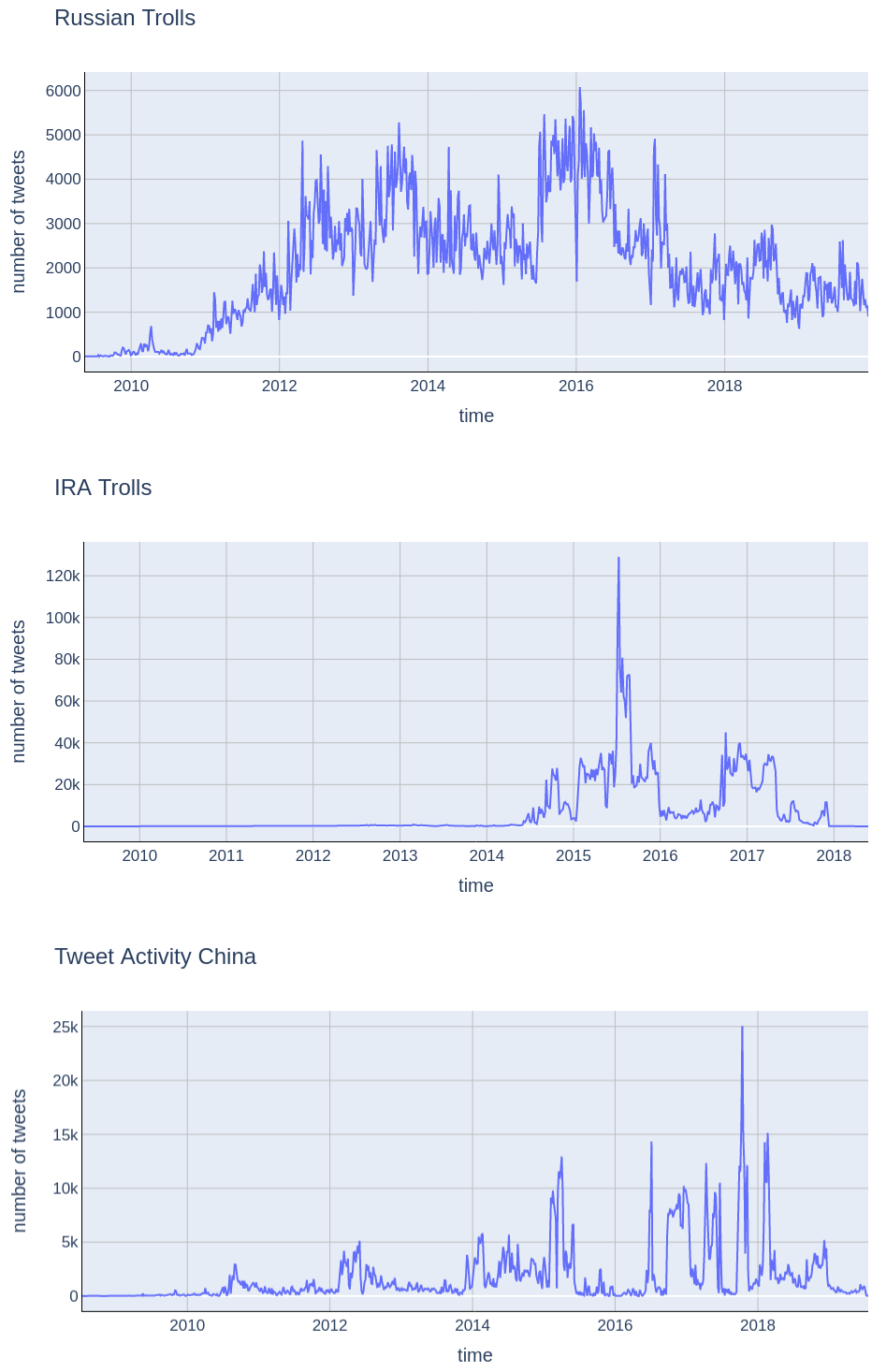


Figure A.4 – Activity generated per time for the Russian (topmost), IRA (middle), and Chinese (bottom) datasets. Such activity is measured by the number of unique tweets generated per time unit. Troll tweets are depicted in blue, while active ones come in red.

Appendix B

Summary of Experiments

B.1 Experiment Settings

Throughout Chapter 4, we have run many experiments in which specific parameters must be set up every time. Figure B.1 presents all those parameters in a tree where traversing from the root to a leaf defines a complete parameter setup. Each experiment begins by defining a data source to extract troll activity from (Russia, IRA, or China), as well as a time interval to do so. Then, we can traverse the tree by setting up the desired parameters.

B.2 More on Link Prediction

In order to take a closer look at how our approach with respect to each experiment setting (Figure B.1), we breakdown our link prediction accuracies per sender (troll of active users), as well as, per source of active users. Figure B.2 shows these accuracy distributions via violin plots only for Russian-based experiments. For completeness, we have included results for our data size *balanced setting* (upper plot), as well as for the setting where we do not place any size restriction (lower plot). It is worth recalling that our balanced setting consists of making the number of receivers proportional to senders and taking a maximum of $5K$ links per experiment.

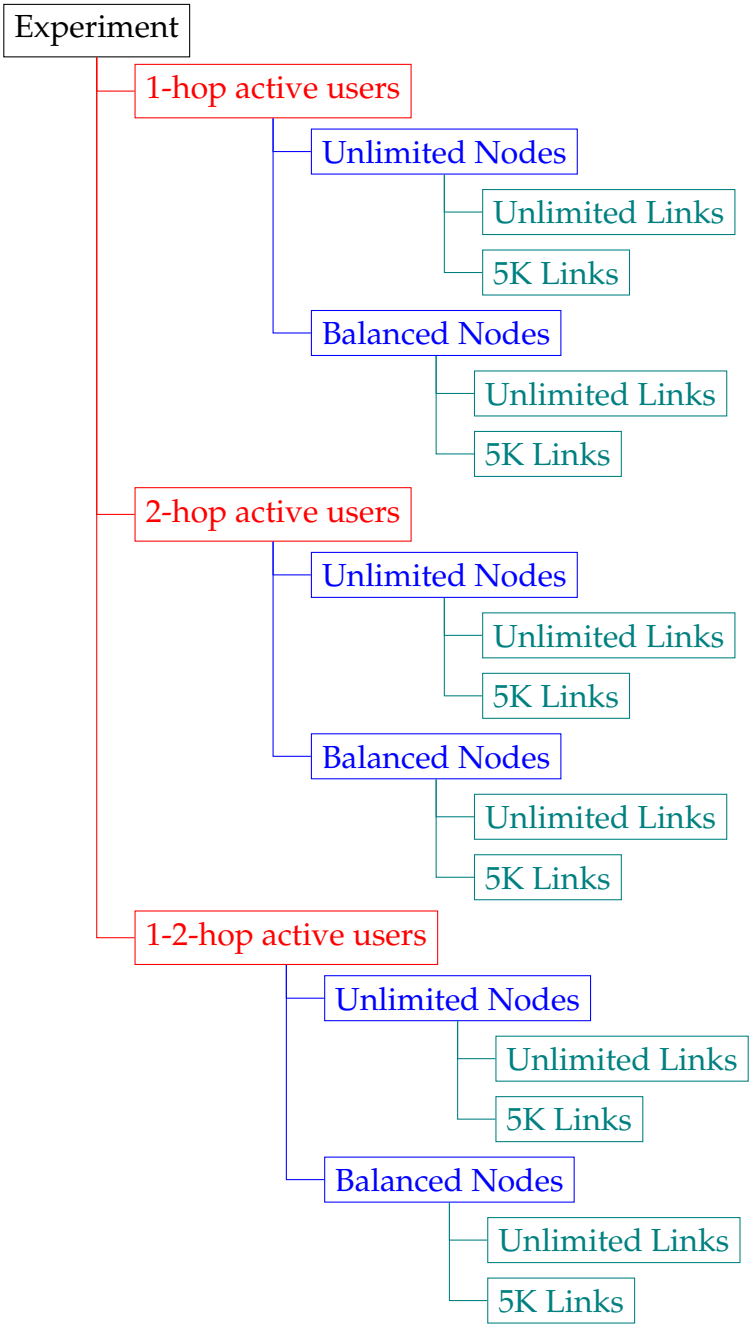


Figure B.1 – This tree illustrates the variety of settings that define every experiment performed in this thesis. Each node defines a parameter to be controlled.

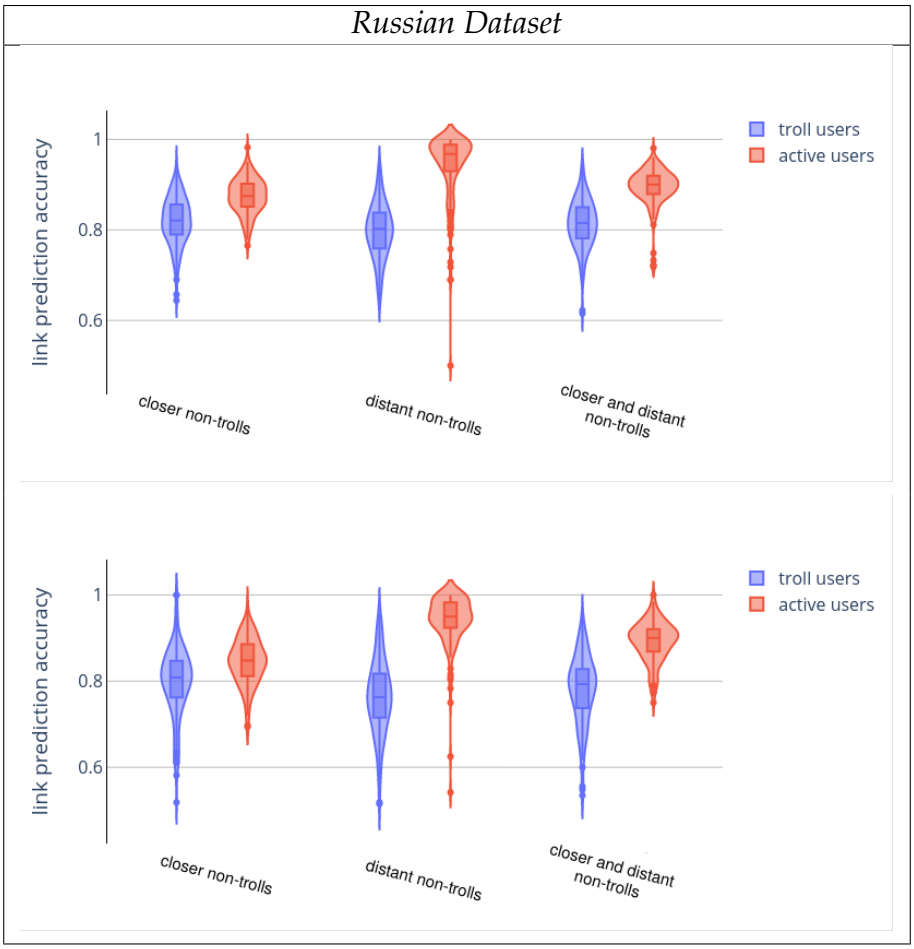


Figure B.2 – Link prediction accuracy breakdown for Russian-based experiments. The top plot displays quantified accuracies for the balanced setting, described in Section 4.1, while the bottom plot provides unbalanced setting's accuracies.

Figures B.3 and B.4 display the same experiment results, as explained above, for the IRA and the Chinese dataset, respectively. Overall, we see that higher non-troll predictability prevails for all Russian and IRA experiments, while Chinese trolls result slightly more predictable otherwise. For the latter dataset, we see that *balancing nodes and links* helps reduce per-class accuracies. Both classes' scores tend to present higher variance (especially trolls) if no restriction is imposed. This fact can be observed by looking at the increase in a score distribution's set of outliers. An opposite phenomenon is observed for IRA distant active users; on the other hand, all troll accuracies tend to have less variance for balanced settings.

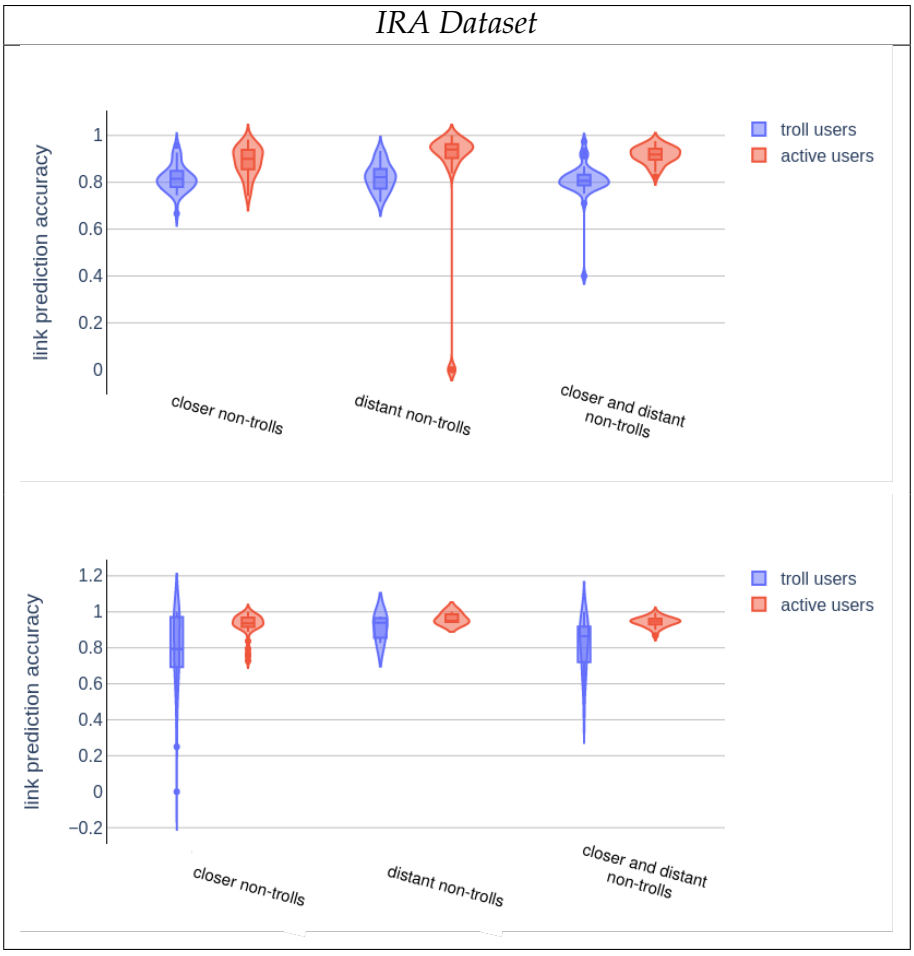


Figure B.3 – Link prediction accuracy breakdown for IRA-based experiments. The top plot displays quantified accuracies for the balanced setting, described in Section 4.1, while the bottom plot provides unbalanced setting's accuracies.

B.3 More on Node Classification

To complement the user-type classification experiments discussed in Section 4.3, we now present validation scores, yet measured by accuracies. The F1 scores in Table 4.4 are more informative, as they leverage false positives, which may be significant in practice, especially in any situation where a platform has to decide on suspending certain accounts. Yet for the sake of further contrasting what pipeline learns from any input, Table B.1 shows that we can provide high validation scores. In fact, for any time interval, the Chinese dataset results in the one with better scores, which are consistently above 80% accuracy.

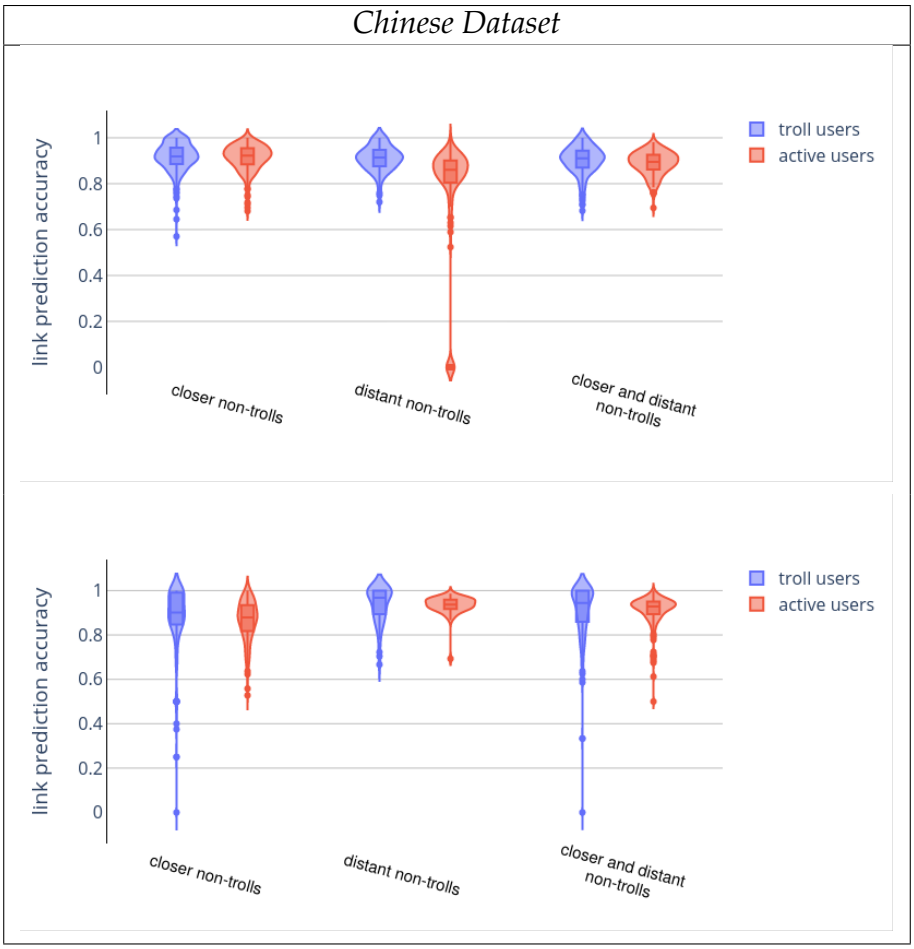


Figure B.4 – Link prediction accuracy breakdown for Chinese-based experiments. The top plot displays quantified accuracies for the balanced setting, described in Section 4.1, while the bottom plot provides unbalanced setting's accuracies.

It is unclear, otherwise, which time interval would be optimal, as maximum scores vary per the choice of active user set.

Lastly, to complement the ablation discussed in Section 4.2, we present validation results with models that have ignored the SEAL framework's node labeling algorithm (see Section 2.4.1 for further details). Table B.2 indicates that the use of node labeling is beneficial, in a small sense, for most experiments. This can be observed from the fact that F1 scores are higher in Table 4.4, which is also true for the link prediction experiments.

Node Classification Accuracies					
Active set	Dataset	5 days	10 days	30 days	60 days
closer	**Russian**	0.77 ± 0.18	0.72 ± 0.16	0.73 ± 0.16	0.77 ± 0.12
	IRA	0.67 ± 0.18	0.68 ± 0.16	0.64 ± 0.16	0.68 ± 0.13
	Chinese	0.80 ± 0.27	0.81 ± 0.17	0.87 ± 0.13	0.83 ± 0.15
distant	**Russian**	0.64 ± 0.29	0.65 ± 0.20	0.65 ± 0.19	0.65 ± 0.17
	IRA	0.70 ± 0.18	0.71 ± 0.17	0.71 ± 0.15	0.69 ± 0.07
	Chinese	0.93 ± 0.18	0.83 ± 0.19	0.86 ± 0.14	0.81 ± 0.13
both	**Russian**	0.81 ± 0.14	0.78 ± 0.14	0.78 ± 0.12	0.77 ± 0.09
	IRA	0.76 ± 0.15	0.79 ± 0.13	0.80 ± 0.12	0.80 ± 0.09
	Chinese	0.87 ± 0.17	0.88 ± 0.12	0.92 ± 0.08	0.85 ± 0.11

Table B.1 – Accuracy Node classification performance scores. We have balanced the number of nodes and links for these experiments to agree with the results presented in Table 4.4.

Node Classification *Node Labeling Ablation*					
Active set	Dataset	5 days	10 days	30 days	60 days
closer	**Russian**	0.52 ± 0.23	0.44 ± 0.15	0.44 ± 0.11	0.47 ± 0.14
	IRA	0.43 ± 0.17	0.44 ± 0.15	0.41 ± 0.08	0.41 ± 0.12
	Chinese	0.69 ± 0.35	0.62 ± 0.29	0.66 ± 0.28	0.56 ± 0.21
distant	**Russian**	0.46 ± 0.26	0.42 ± 0.19	0.41 ± 0.13	0.45 ± 0.16
	IRA	0.41 ± 0.16	0.41 ± 0.15	0.40 ± 0.09	0.40 ± 0.10
	Chinese	0.81 ± 0.28	0.65 ± 0.29	0.68 ± 0.19	0.54 ± 0.20
both	**Russian**	0.54 ± 0.23	0.46 ± 0.13	0.45 ± 0.07	0.46 ± 0.08
	IRA	0.49 ± 0.18	0.47 ± 0.15	0.45 ± 0.07	0.47 ± 0.08
	Chinese	0.79 ± 0.28	0.67 ± 0.28	0.57 ± 0.20	0.50 ± 0.12

Table B.2 – Node classification F1 scores for the ablation setting, where the SEAL framework's node labeling algorithm is ignored.