Supervised feature-based link prediction for shared-content social networks

Pablo Lluch Romero

4th Year Project Report
Artificial Intelligence and Software Engineering
School of Informatics
University of Edinburgh

2020

Abstract

The problem of temporal link prediction has received increasing attention in the last years due to the growing amount of data from social networks. Different approaches have been proposed to address this problem, including supervised, unsupervised, topology-based, non-topology based, etc. In this project, the problem of link prediction is examined for a new class of networks where the non-topological information consists of shared content amongst users. The streaming social network Twitch is chosen as an example for this class of networks to evaluate the algorithms introduced. The link prediction problem is addressed in a feature-based supervised learning approach and a set of topological and non-topological features are examined. The most useful features are found to be Rooted PageRank, Common Neighbours, Betweenness Centrality, shared number of games, common country, and pairwise features regarding the streaming nature of users. Several binary classifiers on these features are explored. Support Vector Machines with Radial Basis Function and Logistic Regression are shown to be the best alternative with an area under the ROC curve of 0.968 ± 0.003. Exclusively using non-topological features resulted in a better performance than exclusively using topological features. Overall, linear and non-linear classifiers resulted in similar performances.

April 24, 2020

Acknowledgements

I would like to thank my supervisor Dr. Rik Sarkar for his attentive guidance and help throughout the project, without whom I wouldn't have been able to develop my interest in social and technological networks. I would also like to thank Benedek Rozemberczki, Rik Sarkar's PhD student, for helping me with idea brainstorming throughout the project and providing me with part of the Twitch dataset used and the code to query it. Finally, I would like to thank my friends, flatmates and family that encouraged me and gave me the strength needed to carry out this project.

Table of Contents

Chapter 1

Introduction

Social networks have exploded in popularity over the past decade. Virtually everyone partakes in them. People share information, form connections and express their opinions. This vast amount of information has unsurprisingly drawn considerable attention from the research community. This is also the case because social networks contain massive amounts of semi-structured data and are detailed representations of human relationships and society.

However, social networks are often incomplete, noisy and ever-evolving. Being able to complete them, understand their dynamics and their evolution patterns is often crucial from a scientific and a business standpoint. A well-studied problem, because of its many applications, is the problem of *temporal link prediction*. That is, given a network of users and their interactions over a period of time, is it possible to predict future interactions between them? Applications of temporal link prediction include friend suggestions, media content suggestion [57], e-commerce [50], collaborator suggestion in scientific publications [33], prediction of new contacts in phone networks [33], and so on.

A wide variety of methods have been proposed to address the temporal link prediction problem. Notable examples include matrix-factorization approaches [38], Markov chains [49], probabilistic models [52] and more [33]. One method that has received increasing attention in the past years is the use of supervised learning on a set of carefully crafted features [43]. These features can be both topological and non-topological and include examples like metrics of neighbourhood overlap [7], shortest distance and its variations [33], degree correlations, clustering–related behaviours, and node content similarity. This has not only shown great performance, but has been able to draw insight into what features are the most essential to predict a link formation, and in consequence, what features and factors most closely speak about the similarity between users and the nature of social relationships.

Nodes (or users) in social networks have a lot of content information. This content information is very different from network to network. It ranges from images, text, articles, to interactions with products or media items. As the impact and the nature of exploitable features differs from network to network, the optimal behaviour of link

prediction methods usually relies on the knowledge of the network and domain, and the correct exploitation of the available data.

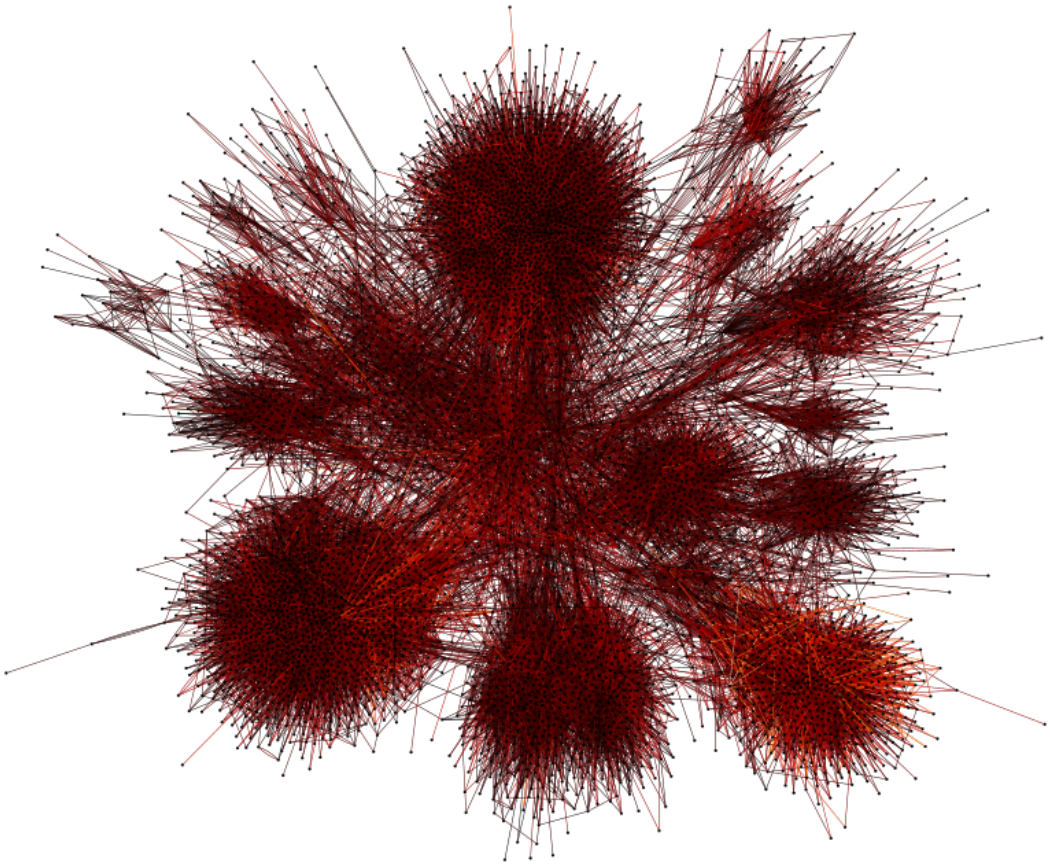

Figure 1.1: Twitch network where vertices represent users, and edges follows between them. The colour of the edges represents the time when they were formed. Darker colours correspond to older dates and lighter colours to recent ones. There are differentiated clusters with similar edge formation dates. The original network was subsampled for ease of visualization.

The content produced by users in social networks is usually unique, in the sense that it doesn't fully fit within a common shared category or type. For example videos in *YouTube* or pictures in *Flickr*. However, this doesn't always have to be the case. In social networks like *Twitch* or *Facebook Gaming*, users produce content that fully fits within a specific category. Most importantly, these categories are fixed and shared amongst all the users. An example of a type of category is video games. In streaming platforms like *Twitch* and *Facebook Gaming*, users are allowed to live-stream about a set of video games that is shared amongst the community. Link prediction in the context of this class of social networks has not been explored before.

This class of social networks poses interesting questions about how the shared content can be modelled and exploited optimally. Being able to classify content in categories

that are shared between users has the potential to build robust similarity metrics between them. Consequently, the aim of this work is to assess how this non-topological shared information and the available topological data can be exploited optimally for this class of networks. In order to do so, the social network *Twitch* was chosen to carry out the necessary experiments.

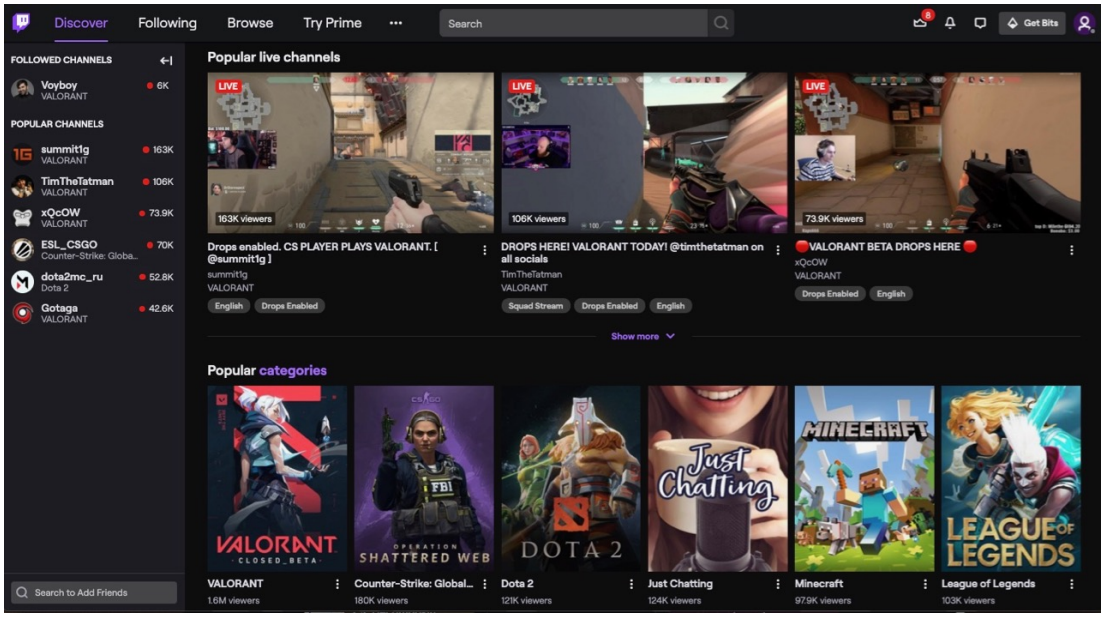

Figure 1.2: Screenshot of Twitch main page [5]. Popular live channels and categories are displayed in the Discover pannel.

Twitch is a streaming platform where users upload and broadcast videos of them playing video games (Screenshot of main page in figure 1.2). Little work has been carried out on Twitch due to its very recent addition to the social streaming scenario. Twitch went from 102K viewers in 2012 to 1.72M in 2020 [6]. Twitch currently dominates the live streaming market with 2,720 million live hours watched, followed by YouTube with 735 million [4]. Figure 1.1 shows a subsampled visualization of the Twitch follow graph.

Another distinctive feature of this family of networks is that users don't form links solely on the basis of similarity, unlike most of the networks where link prediction algorithms have been tested. Factors like node centrality, streaming frequency or genre popularity play a role in the user's decision to follow someone else [25]. Although this is also true in networks like *Twitter* or *YouTube*, the degree to which these features influence the link prediction in conjunction with the novel non-topological features was studied for shared-content networks.

In short, the impact of similarity and non-similarity based topological and non-topological features is explored to produce a supervised link prediction algorithm. This link prediction is assessed for the class of networks where the category of content is shared amongst users.

1.1 Contributions

The work carried out makes the following contributions:

- Explore the impact of non-topological features for link prediction in shared-content networks. The shared number of games, common country and pairwise features regarding the streaming nature of users resulted in the most useful features according to their coefficient weights in binary classifiers (weights of at least 0.7 in all classifiers).
- Explore the impact of topological features for link prediction in shared-content networks. Rooted PageRank, Common Neighbours and Betweenness Centrality were the most useful features according to their coefficient weights in binary classifiers (weights of at least 0.8 in all classifiers).
- Explore a range of different latent embedding methods and distance functions to exploit the non-topological information of shared-content networks for the problem of temporal link prediction. Non-Negative Matrix factorization paired with cosine similarity was found to be the most effective approach to exploit the shared-content information with an area under the ROC curve of 0.701 ± 0.004. Non-Negative Matrix factorization was seen to generate sparse embeddings that correspond to interpretable video game genres.
- Explore the performance of binary classifiers for link prediction in shared-content networks. The classifiers explored are Linear SVMs, RBF SVMs, Perceptron and Logistic Regression. Logistic Regression and RBF SVM resulted in the best performance with an area under the ROC curve of 0.968 ± 0.003. Linear SVMs resulted in the best performance when only topological features were used. Exclusively using non-topological features resulted in a better performance than exclusively using topological features

1.2 Report structure

Here the structure of the report is detailed.

- Chapter 2 introduces the relevant theoretical and contextual background: previous work on Twitch, supervised link prediction, collaborative filtering, and embedding algorithms are introduced.
- Chapter 3 discusses the nature, obtaining and processing of the Twitch dataset. It also establishes the problem formulation and the experimental setup.
- Chapter 4 discusses the choice of a set of commonly used and novel topological features and visually evaluates their impact.
- Chapter 5 discusses the choice of non-topological features and visually evaluates their impact. It also experiments with the use of multiple embedding methods and distance measures to exploit the non-topological features for link prediction.
- Chapter 6 combines non-topological and topological features to create a link

predictor. Different binary classifiers are tested and the impact of different features is assessed. Insight is thrown into how streaming behaviour and network topology affects the formation of links.

Chapter 2

Background

This chapter consists of a comprehensive introduction to Twitch, the problem of supervised feature-based learning for link prediction and the necessary theoretical information to support the work carried out.

2.1 Twitch

The ability to stream live content has become an increasingly popular feature in social networks like Instagram, Facebook and YouTube. Companies have released dedicated live-streaming sub-platforms like YouTube Live, Facebook Gaming and Microsoft's Mixer. However, there is one platform that has been dominating the live streaming scenario for a few years now: *Twitch*. Very recently, the gaming community saw a shift of users wanting to not only play video games but see others play them. This shift was partially motivated by the new notion of e-Sports (competitive video games) and official tournaments.

Twitch is a streaming platform where users upload and broadcast videos of them playing video games. Although most of the users stream game-related content, there is a small community of users that stream other content like video blogs, tutorials or cooking classes. The work carried out focuses on the games.

Each user has a corresponding channel that goes live whenever the player is broadcasting. However, not every user is a streamer. Twitch has over 15 million daily active users and between 2.2 and 3.2 million monthly broadcasters [1].

The only research that has been done on Twitch is statistical analysis. Deng et al. 2015 [16] study the viewing distribution across games to uncover tendencies, viewing patterns and game ecosystems. Hamilton et al. [25] explore viewer participation on streams and describe how stream communities form and what motivates members to join these communities.

There are also a few examples where Twitch's data has been used to evaluate graph-based algorithms. Rozemberczki et al. [47] use a small portion of Twitch's available data to evaluate a network embedding algorithm. However, no work has focused on

Twitch to design predictive algorithms.

2.2 Link prediction

The evolution of social networks is a complex process guided by many interactions, dynamics and hidden parameters. Understanding these dynamics can be a hard problem. A simpler task is to limit the study to the interaction between pairs of nodes.

Link prediction can be considered in a static or dynamic setting. In a static setting, the problem of link prediction consists of identifying missing links between users that might have an interest to connect. Online social networks have inherently missing information. They are only partial representations of real-world human interactions that contain an incomplete subset of people's identities and actions.

In a dynamic setting, link prediction tries to predict the evolution of a social network by predicting the edges that will appear or disappear in a given time frame. This specific problem is called temporal link prediction. For the rest of the report, whenever the term *link prediction* is used it will always refer to the temporal version of it.

The problem of link prediction relies on obtaining a score between pairs of nodes that determines the likelihood of an edge forming between them in the future. Two main approaches have been proposed to this problem: *similarity-based* and *learning-based*.

In the similarity-based approach, the notion of proximity is used to calculate a similarity score between pairs of nodes [53]. However, finding the right features to describe this proximity is not trivial and the right selection and weighting is usually network-specific. For that reason, a learning-based approach has been common in the recent literature. In the learning-based approach, the link prediction problem is framed as a binary classification task [8]. Pairs of vertices are used as instances described by a set of features. The labels represent whether a link exists between them or not.

2.2.1 Supervised feature-based learning for temporal link prediction

The question of link prediction is very closely related to node similarity. The social principle of homophily has been shown to be present in many social networks [37], so it's a well-known assumption that the more similar two users are the more likely it is for an edge to exist between them. One could extend this argument and claim that the neighborhood of a node is an alternative way to describe a user. This led to the development of link prediction algorithms where both node-specific and neighborhood related features are considered.

Clearly, content, topological and social features are very different in shape and form. It would be hard to manually quantify how relevant each of the individual features are to predict the presence of a link. This motivated the decision of using learning-based methods to combine them all. In using learning-based methods, specifically feature-based classification, machine learning models are able to capture interrelationships between different features, and their impact on the link formation.

Consider $a, b \in V$ to be nodes of the Graph $G(V,E)$. Consider $l_{a,b}$ to be a value that encodes whether the edge (a,b) is in E or not in the following way:

$$l_{a,b} = \begin{cases} +1 & \text{if } (a,b) \in E \\ -1 & \text{otherwise} \end{cases} \tag{2.1}$$

The problem of link prediction could be framed as a binary classification problem where the attributes are a set of cautiously constructed features. The binary classifier aims to learn a similarity metric between nodes, and use it to estimate the likelihood of a link forming in the future.

Unlike other methods of link prediction, supervised feature-based learning has benefited from the explicit and well-defined nature of features. Liben-Nowell et al. [33] explore the impact of different topological features for link prediction in citation networks. Al Hasan et al. [8] show that non-topological features help the link prediction problem using supervised learning considerably. They use decision trees, k-nns, multilayer perceptrons and SVMs as binary classifiers.

Lichtenwalter et al. [34] provide a general framework for supervised link prediction in sparse networks. They consider issues such as network observational period, generality of existing methods, degrees of imbalance and sampling approaches and address them in a collective manner. They evaluate their methods on a cellular phone network and a network of physics collaborators.

Different choices of binary classifiers have been explored. From linear to non-linear, examples of classifiers explored in feature-based link prediction are Decision Trees, RBF Networks, Naive Bayes, Perceptrons, and Support Vector Machines [15] [20] [8] . In this work, three linear classifiers and a non-linear one are explored: Support Vector Machines, Logistic Regression, Single Layer Perceptron, and Support Vector Machines with Radial Basis functions, respectively. The following sections give a high level explanation of these classifiers.

2.2.2 Support Vector Machines

Support Vector Machines (SVM) are used in this work as they have shown to be successful in many feature-based link prediction applications [15] [20]. Al Hasan et al. 2006 [8] show SVM + Radial Basis Function kernel to be the best performing binary classifier for a set of topological and non-topological features amongst other classifiers like Decision Trees, Naive Bayes or Multilayer Perceptron.

SVMs, originally introduced as Support vector Systems [14] are a learning algorithm intended for binary classification. SVMs work by producing a decision boundary –in the form of a hyperplane– that separates the original data in the desired classes so that the distance between any point and the decision boundary is maximized.

The decision boundary is obtained by solving the optimization problem [22]:

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 s.t. y_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq +1 \text{ for all i} \tag{2.2}$$

Where $\mathbf{w}$ represents the coefficients of the hyperplane, $\mathbf{x}$ is a vector encoding the dimensions of each datapoint and y a binary encoding of the class of each datapoint. Regularization has been commonly used for SVM to avoid overfitting. By introducing a regularization term C that penalizes high values of $\mathbf{w}$, equation 2.2 can be rewritten as:

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 + C\sum_i \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0)) \tag{2.3}$$

Where small values of C result in stronger regularization.

SVMs were originally intended for linearly classifiable data. Linear SVMs offer the ability to quantify the impact that individual features have on the classification. However, SVMs have been successfully extended to classify non-linearly separable data through the use of kernel functions. Kernel functions work by transforming the original data into a new dimensional space where the previously non-linearly-separable data becomes linearly separable [14]. A common kernel is the Radial–Basis function (RBF) where datapoints are transformed by calculating their distance to a fixed point.

Solving equation 2.3 relies on a linear algorithm that depends only on $\mathbf{x}^T\mathbf{x}'$. Consequently, there's no need to transform every instance $\mathbf{x}$ but rather every pair $\mathbf{x}^T\mathbf{x}'$. This is called the kernel trick and allows kernels to be defined for pairs of data that results in computational speed-ups [22]. Using the kernel trick, the RBF kernel can be defined as:

$$k(\mathbf{x}, \mathbf{x'}) = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2) \tag{2.4}$$

Where $\gamma > 0$ controls the influence of individual training datapoints in the calculation of the weights [44].

2.2.3 Logistic Regression

Logistic Regression (LR) is a model for linear classification where the probability of each sample belonging to either class is modelled using a logistic function [27]. LR has been used for supervised link prediction where path-based features from multiple source networks were exploited [36]. LR has also been used for link prediction in heterogeneous networks to exploit social patterns[17].

LR separates the data by fitting a hyperplane given by parameters $\mathbf{w}$ so that the probability of the data belonging to the right class is maximized, i.e. for data X and labels Y, maximize $P(Y|X = x)$. This is equivalent to maximizing the log likelihood of the data (D) given the parameters $\mathbf{w}$:

$$log(p(D|\mathbf{w})) = \sum_{i=1}^{n} log(\sigma(\mathbf{w}^T\mathbf{x}_i) + (1 - y_i)log(1 - \sigma(\mathbf{w}^T\mathbf{x}_i)) \tag{2.5}$$

Where σ is the sigmoid or logistic function $\sigma(z) = \frac{1}{1+e^{-z}}$, $\mathbf{x}$ is a vector that encodes the dimensions for each datapoint and y a binary encoding of the label of each datapoint.

2.2.4 Single Layer Perceptron

A single Layer Perceptron is a model for linear classification where the class of a datapoint (y) is given by [22]:

$$y = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + \mathbf{w}_0 \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.6}$$

Where $\mathbf{w}$ is the learnt weight vector and $\mathbf{x}$ a vector that encodes the dimensions of each datapoint. The single layer perceptron is the building block of artificial neural networks by being arranged in multiple layers that increase the complexity of the model. However, the simplicity of the Single Layer Perceptron allows interpretability of the importance of each original feature.

2.2.5 Beyond similarity

The directedness of links is something that has been rarely considered for link prediction, especially in the supervised feature-based learning approach. Undirected research publication networks like *arXiv*'s *astro-ph* for astrophysics, BIOBASE [2] for biology and DBLP [3] for Computer Science have been the default datasets to evaluate link prediction algorithms.

The symmetry in undirected networks allows the edge prediction to be made solely on the basis of node similarity. However, for directed networks, asymmetrical attributes that capture the directionality of the edge need to be introduced. Valverde et al. 2013 [51] explore the problem of supervised link prediction for a directed network, Twitter. They exploit community behaviours to improve the prediction. They however don't address the problem of link prediction in a feature-based setting so they don't investigate the choice and impact of features.

Cukierski et al. 2011 [15] evaluate 94 topological features for the problem of supervised feature-based link prediction using Random Forests as a classifier. They explore the effect of these features on the directed social network *Flickr* as part of *The 2011 IJCNN Social Network challenge* and explore the performance when each feature is used exclusively to predict a link. Fire et al. 2011 [20] also explore the use of topological features for directed and undirected social networks like Facebook and YouTube.

The work carried out in this project explores the use of topological and non-topological features, so the next sections will introduce the necessary background for the proper exploitation of topological and non-topological information.

2.3 Topological features

Most of the metrics discussed below are explained for the undirected setting as that's how they were originally described. Besides, they can be easily extended to the directed setting through consideration of what directionality means in the pertinent network. That will be discussed in section 4.

2.3.1 Neighbourhood based metrics

Let $\Gamma(x)$ be the set of neighbours of x in an undirected setting. For the directed setting, let $\Gamma_{out}(x)$ be the set of out-neighbours and $\Gamma_{in}(x)$ the set of in-neighbours.

In the traditional setting of link prediction where links happen solely on the base of similarity, numerous metrics have been proposed that use neighbourhood overlap as a metric of similarity. This has been shown to be true in multiple settings, most notoriously in citation networks [33] [7].

Common Neighbours (CN): The most straight forward approach is the number of common neighbours, shown to be an effective predictor of collaboration in citation networks [41]:

$$CN(x,y) = |\Gamma(x) \cap \Gamma(y)| \tag{2.7}$$

Jaccard's Coefficient (JC): Salton & McGill, 1983 [48] introduced the *Jaccard coefficient* as a normalization of the common neighbours based on the total number of neighbours of x and y:

$$JC(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{2.8}$$

Adamic-Adar Coefficient (AA): Adamic et al. 2003 [7] refined the previous features by giving a higher weight to *rarer* neighbours:

$$AA(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{log|\Gamma(z)|} \tag{2.9}$$

2.3.2 Distance based metrics

A very intuitive way to capture similarity between two vertices is the notion of *distance* in the network. Many approaches have been proposed to capture this notion of distance, with different levels of complexity.

Shortest path (SP): The most straight-forward definition of distance is the length of the shortest path between two vertices. Even though effective for its simplicity, *shortest path* is a naïve metric. This is the case due to the *small-world* problem [41] of social networks, where every pair of nodes is connected through a short chain of edges. Even though this concept is essential for the ability of content to quickly spread across networks, it makes vertices that would be considered to be far away seem like they're closely connected. Another problem of the *shortest path* metric is that it's not robust to a small set of edges bridging distanced components. Multiple methods have been proposed to address these two concerns. One that's been widely used is *Rooted PageRank*.

Rooted PageRank is an alteration of PageRank [12], which is the backbone of ranking algorithms in many Information Retrieval applications. From a source vertex x, the PageRank of another vertex y is the probability of a random walk to reach it. In order to *punish* distant nodes, PageRank introduces a fixed probability α of resetting the random walk at each step. The final score of each vertex y can be thought as a distribution of an initial weight that started in x.

2.3.3 Social theory based metrics

Social networks are a reflection of society. Consequently, behaviours that motivate the formation of relationships in human groups also translate to social networks. This involves principles like homophily, weak and strong ties, triadic closure, centrality and community clustering. Centrality and clustering will be discussed in more detail below.

2.3.3.1 Centrality

Li et al. 2011 [32] show the *centrality* of nodes to be a determining factor in link prediction, i.e. users not only choose who to follow on the basis of similarity but are more likely to follow central nodes.

Centrality is a measure of the importance of vertices in a network. The definition of *importance* is variant and domain-specific. The methods proposed to calculate centrality are accordingly varied too. A common way to define importance is based on the amount of *flow* a node carries [10]. A naïve way to capture this notion of flow is the in and out degree of the node. More advanced techniques have been proposed, namely *betweenness centrality* and *PageRank*.

The **Betweenness Centrality** of a vertex is proportional to the number of shortest paths that go through that vertex. Betweenness centrality could be also thought as a measure of the control over the information flow that a node has over a network [42]. Liu et al. 2013 [35] show betweenness centrality, amongst other features, to be a determining feature in the formation of links.

PageRank is a generalization of *Rooted PageRank*, as explained in section 2.3.2, where every node is a source node, i.e. the initial weight distribution is uniform across all nodes. PageRank is especially useful in directed graphs, where a forward edge is interpreted as a way to delegate importance in another node.

2.3.3.2 Clustering

Clustering Coefficient (CC): How clustered a node is has shown to be a descriptive feature for vertices in social networks and the task of link prediction[33]. Vertices in denser communities are more likely to form new links.

Newman et al. 2001 [41] proposed a local clustering coefficient $CC(x)$ for a vertex x defined as:

$$CC(x) = \frac{|\{e_{jk}\}|}{|\Gamma_{out}(x)|(|\Gamma_{out}(x)-|1)} : v_j, v_k \in \Gamma_{out}(x), e_{jk} \in E \tag{2.10}$$

Where E is the set of edges of the graph involved.

2.4 Non-topological features

Nodes in social networks represent users, and these users are not blank vertices but are usually filled with content and information. However, the nature of this content varies

greatly. Especially in online social networks where users can post and create a wide variety of content. Examples of different types of content are text documents, images, videos, demographic information, etc.

The nature of this content is usually very high dimensional. E.g. text or images, where the number of dimensions is the size of the vocabulary or the number of pixels, respectively. This high dimensionality results in problems like the curse of dimensionality [21]. It becomes impossible to compare individual dimensions. As the number of dimensions increase, the amount of data stays the same, so the data becomes rapidly sparse. A method called *feature embeddings* has been widely used in the literature to address this problem.

2.4.1 Feature embeddings

An embedding is a representation of an object –classically represented by a vector– in a lower-dimensional latent space. The goal of an embedding is to capture semantic meaning about objects so that similar objects end up close in the latent space.

Embeddings have seen a recent increase in attention due to the boost in data digitalization and the surge of big data. These bigger datasets have powered the use of Collaborative Filtering through embedding methods for recommender systems. For applications where users interact with a collection of items, Collaborative Filtering captures underlying user preferences by understanding the user-item interactions.

Given a matrix $X(\text{n x m})$ where n is the number of items in the dataset and m the dimensionality of those items, the goal of an embedding is to produced a new matrix $X'(\text{n x m'})$ where $m' < m$. A *quality* embedding serves both as a dimensionality reduction and as a feature extraction, i.e.:

- It helps reduce noise
- It reduces the size of the data for easier processing
- It uncovers user-centric latent features
- It uncovers relationships between features

In par with the varied nature of the possible content, a variety of methods have been proposed to generate embeddings. The following sections provide a theoretical explanation of the embedding methods that were used in this work.

2.4.1.1 Truncated SVD and PCA

The Singular Value Decomposition (SVD) of a matrix A is a matrix decomposition of the form:

$$A = U \cdot D \cdot V^T$$

where U and V are orthogonal matrices and D is a diagonal matrix [23].

Truncated SVD is a natural extension of SVD where only the k largest singular values are calculated.

$$A \approx A_k = U_k \cdot D_k \cdot V_k^T$$

Truncated SVD is a very common techniques for producing low dimensional embeddings of high dimensional data. The newly generated features consist of a linear transformation of the original components. These features are such that maximize the variance of the transformed data.

A very similar technique to truncated SVD is Principal Component Analysis (PCA) [54]. The difference between PCA and SVD is that for PCA, the input data A is first centered and often reescaled so that each original dimension has variance and mean 1. This is very helpful because the dimensions of real data don't usually have the same range. PCA is commonly used to create interpretable representations of high-dimensional data.

On the flip side, when A is a sparse matrix, standardizing it results in a lot of unnecessary memory, as most of the entries in X are zeros. Conversely, SVD is very suitable for large sparse matrices. Not centering the data prior to the factorization allows the use of sparse matrices, where only the non-zero entries are used.

2.4.1.2 Non-negative Matrix factorization

Non-negative matrix factorization (NMF) is another type of matrix factorization where the original matrix and its decomposition is restricted to only take positive values. This kind of factorization takes the form:

$$X \approx H \cdot W$$

Where the dimensions of X,H,W are (n x m),(n x k) and (k x m) respectively. k is the number of components (new dimensions that the original data is reduced to).

This factorization is obtained through a numerical approximation where a distance between X and the factorized product HW is minimized. A common way to calculate this distance is using the Frobenium Norm, which is an extension of the Euclidean distance to matrices:

$$d_{Fro}(X,Y) = \frac{1}{2}||X-Y||^2_{Fro} = \frac{1}{2}\sum_{i,j}(X_{ij}-Y_{ij})^2 \tag{2.11}$$

However, different distance metrics have been proposed such as the Kullback-Leibler or the Itakura-Saito [19].

The non-negativity constraint allows a more natural factorization of matrices X of real data. Applications of NMF for natural positive data like images, text or ratings have recently been very successful. A well-known case is the success of NMF for recommender systems in the *Netflix Price Competition* [28]. An intuitive explanation of this success is that by restricting a factorization to only use additions, you are creating a set of components that collectively form a whole. The addition of sparseness constraints has been proven to be especially useful to create *parts-based representations* [26].

A problem with NMF is that it is prone to overfitting. Overfitting occurs when the distance between the matrix factorization and the original matrix is very small but the underlying structure of the data is not captured. This can happen due to the exploitation

of noise. Overfitting is usually stopped by regularization. Regularization consists on limiting high values in H and W, as high values wouldn't normally arise from decompositions of natural data. When regularization is considered, the objective function to be minimized is:

$$d_{Fro}(X,WH) + \alpha\rho||W||_1 + \alpha\rho||H||_1 + \frac{\alpha(1-\rho)}{2}||W||^2_{Fro} + \frac{\alpha(1-\rho)}{2}||H||^2_{Fro} \quad (2.12)$$

Where ρ controls how much an element-wise penalty (L1) vs a Frobenius-norm penalty (L2) is used, and α controls the intensity of the regularization.

Choosing a distance measure between the resulting embeddings to generate a similarity metric is not a trivial task for NMF. Xue et al. 2014 [55] evaluate the impact of different distance measures for a variety of tasks and suggest a set of custom distance metrics that can be particularly effective.

They introduce two variations of a custom distance metric for NMF-based classification and pattern recognition: non-negative vector similarity coefficient-based (NVSC) that exploits the nature of the NMF embedding. The two variations of the distance function are:

$$d_{NVSC1}(X,Y) = \frac{\sum_{i=1}^{n} min(x_i,y_i)}{\sum_{i=1}^{n} max(x_i,y_i)} \qquad d_{NVSC2}(X,Y) = \frac{\sum_{i=1}^{n} min(x_i,y_i)}{\sum_{i=1}^{n} (x_i+y_i)/2} \quad (2.13)$$

2.4.1.3 Latent Dirichlet Allocation

Topic modelling addresses the problem of extracting the topic or topics of a set of documents. Consequently, a document can be embedded by the vector encoding the probability of each topic being present in it.

A commonly used algorithm for topic modelling is Latent Dirichlet Allocation [9]. LDA is a generative probabilistic model that uses a three-level hierarchical Bayesian model. Documents are modelled as a finite mixture of a set of topics. The topics themselves are also modelled as a mixture of sub-topic probabilities.

The fact that topics are modelled by a mixture of sub-topic probabilities is useful when documents don't have a unique topic. This is highly beneficial in the streaming problem, as we are trying to uncover a combination of latent user preferences in the context of LDA called topics, and not identifying each document to a single topic/preference.

Because LDA is usually used for text-based data, a TF-IDF encoding of the documents is usually adopted [29]. TF-IDF, short for term frequency–inverse document frequency, is a common way to represent text in information retrieval applications [45]. It's beneficial because terms are weighted based on how common they are. Very popular terms, e.g. stop words, are weighted down while rarer, more meaningful terms are weighted up.

2.4.1.4 Doc2Vec

The last embedding method I explored was Distributed Representations of Sentences and Documents (Doc2Vec) [30]. Doc2Vec is an unsupervised algorithm that learns

fixed-length representations of documents or paragraphs. Doc2Vec is an extension of Word2Vec [40] where the representations are created for words instead of documents.

Both Doc2Vec and Word2Vec are based on the skip-gram model to generate embeddings based on the contex a word appears in. The context of a word in Doc2Vec could extend up to the whole document. That way it's possible to use Doc2Vec in scenarios where word ordering is not important, and treat documents using the *Bag Of Words* (BOW) model. In this scenario, a variation of Word2Vec is proposed called the Distributed Bag of Words version of Paragraph Vector (PV-DBOW).

The goal of the skip-gram model is to maximize the predicted probability of a word given its context. In order to do so, a softmax function is used. Word2Vec and Doc2Vec approximate this softmax function using two alternatives: Hierarchical Softmax [39] and Negative Sampling [40].

LDA uses TF-IDF encoding to deal with the excessive impact of popular words, as discussed in section 2.4.1.3. The way Doc2Vec addresses this problem is through changing the sampling distribution. This distribution is parameterized by a constant known as the *negative sampling exponent* (NS exponent). A value of 1 samples according to the original word frequencies, while a value of 0 performs a uniform sampling where all words are sampled with equal probability. Negative values sample low-frequency words with a higher probability [46].

Chapter 3

Dataset, problem formulation and experimental setup

A source of non-inherent missing information in social networks is the fast pace at which these massive networks evolve. Obtaining complete information for large temporal sequences is extremely expensive, historical topological information is hardly ever available through APIs, and extensive querying over a large time frame is expensive –time and resource wise. The timestamp of the link formation in mainstream social networks is usually not publicly accessible information. Previous work has addressed this problem by masking a portion of the existing edges and try to predict them as if they were generated in the future. However, this set up is problematic when the evolution of the social network is affected by the same links that are being masked.

Obtaining non-topological historical information is usually a very limited task for mainstream social networks too. Public APIs usually limit for the amount of temporal non-topological information that can be accessed by the public. However, Twitch's API allows the querying of its follow graph with timestamped link formation. Twitch's API also allows a good amount of historical non-topological information to be queried.

In order to compile the dataset needed to carry out the presented work, the Twitch public API was queried twice:

- In May 2018 to obtain the full Twitch social network consisting of user information, their stream history from 2011, and timestamped follow links between users accurate to the query date. The querying of this data was done by Benedek Rozemberczki, a PhD student of my supervisor Dr. Rik Sarkar.
- In January 2020 to obtain the timestamped follow links between the users found in the 2018 query. This portion of the data was queried by me.

The full Twitch follow graph contains more than a million nodes. However, the data was filtered to reduce its size while preserving its variability. Users in Twitch could be put into one of three groups: pure streamers, pure viewers and viewer-streamers. From a research perspective, the latter is the most interesting group as they both watch content from other users and upload their own content. Only viewer-streamer users are

considered. US streamers were not included to reduce the dataset size while preserving most of the country variability.

3.1 Nature of topological information

Figure 3.1 shows the distribution of dates of the edges in the dataset. The true number of follows per date bin is exponential up to the current date. However, the edges in *test edges* are only edges that formed between the nodes in the train edges. As time goes on, new users are introduced in the real graph that this dataset doesn't account for. Besides, the original set of users in the train set become less active over time. That is why the distribution shows a decaying tendency after the date of the streaming crawl.

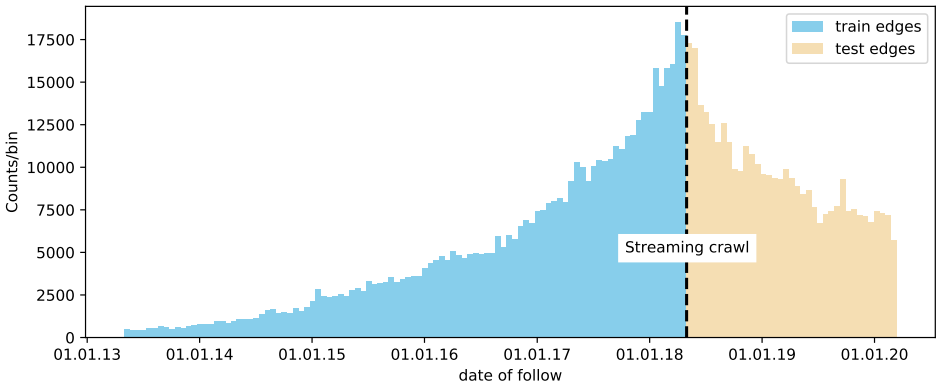

Figure 3.1: Date of follows in the dataset. The rate of edge formation increases exponentially until the date of the streaming crawl. Due to how the data was obtained the rate decreases after that.

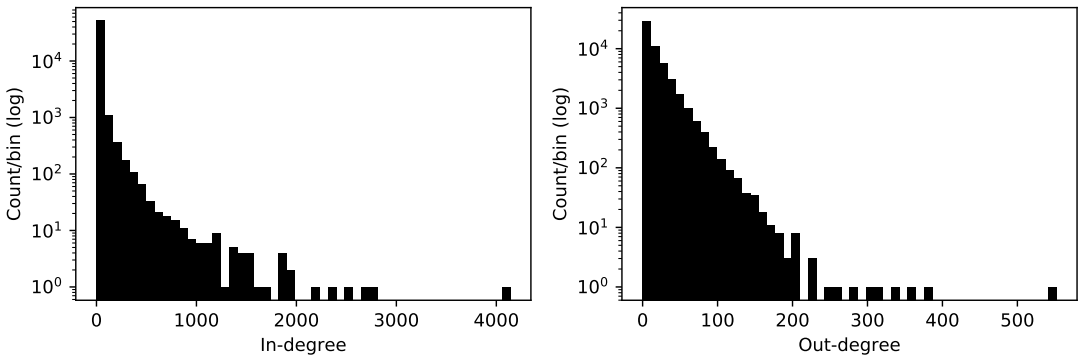

Figure 3.2: In-degree and out-degree distribution across the vertices. In-degree is more polarized while out-degree is more uniform.

Figure 3.2 shows the degree distribution of nodes in the graph. The out-degree (number of followees) is more uniformly spread than the in-degree (number of followers). The majority of users have a small in-degree (close to 0) while a small portion of users (famous relevant streamers) have a very high in-degree. This is consistent with Deng

et al. 2015 [16]'s findings that shows that a small subset of streamers dominate the Twitch scene.

3.2 Nature of non-topological information

The dataset includes demographic information about each user and information about the streaming history of each user. The demographic information consists of:

- Whether or not the user is a partner
- The language of the user
- The country of the user
- The date when the user created their account
- Whether the user produces mature content
- Number of aggregated views

The information about the streaming history of each user contains the following relevant information for each streaming:

- Name of the game: given as a string by the user. It's thus prone to different spellings, grammar mistakes, and different choice of capitalization
- String description of the streaming
- Number of views
- Publishing date

The fact that the name of the game is given as a string limits the ability to know when two streams correspond to the same game. As a first step, the string was converted to lower case to avoid capitalization mismatches. The use of feature embeddings was used as a more extensive approach. This is discussed in section 5.2.2.

For each user, the dataset in question contains information about the video games they played in the past. The amount of information available for each user varies greatly based on several reasons:

- Users are given the option to save highlights of their favorite video games for an unlimited amount of time.
- Users are given the option to save (archive, as called in Twitch) the content that they streamed as videos in their channel for a limited amount of time. This amount of time changes based on the type of user. Partners are able to archive their videos for 60 days, while regular users only for 14.

The access to the videos through the API is possible as long as the videos are available in the user's channel.

As it can be seen in figure 3.3, there is a drastic difference in the amount of information before and after 60 days prior to the dataset collection. There is also a slight drop 14

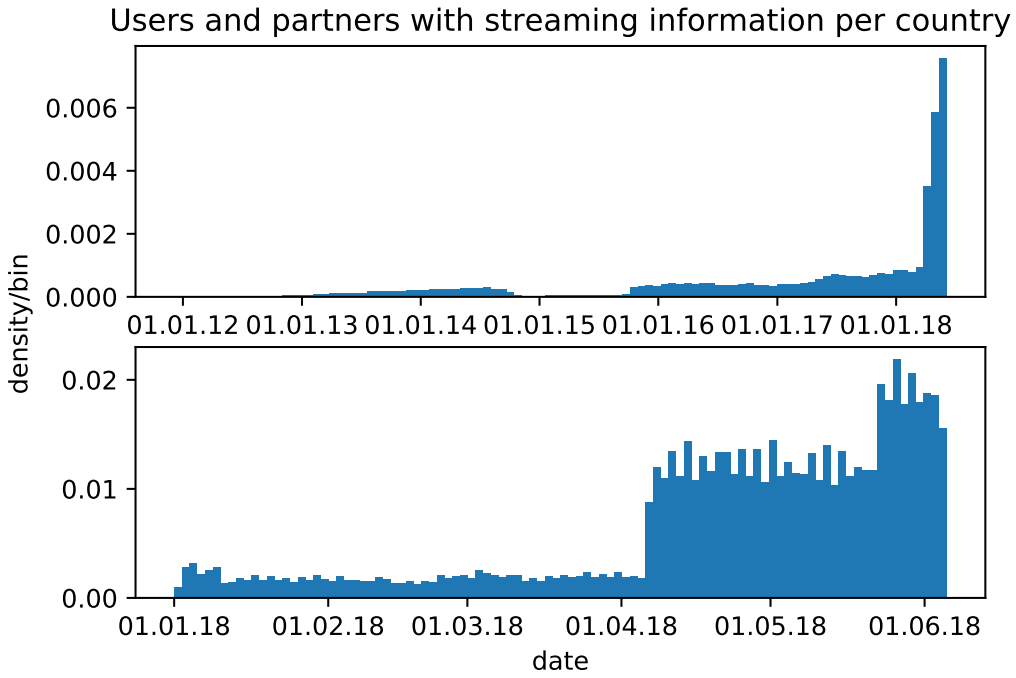

Figure 3.3: Dates of streams. A 14-day and 60-day drop can be seen due to the temporal storage nature of Twitch.

days prior to the dataset collection. The 14 day drop is not as drastic as the 60 day one. This is because partner users not only can archive their videos for 60 days but they are much more active in the platform so they generate a larger amount of streaming information.

Figure 3.4 shows the skewed distribution of video games. It's important to notice that the x axis is being represented in a log scale as the majority of the games have a relatively small number of players. It's interesting to notice that the 50% of the players have played at least one video game in the top 165 most popular video games (0.4% of all games). Also, It's interesting to notice that the 90% of the players have played at least one video game in the top 4,500 most popular video games (11.7% of all games). This means that the most popular video games contain most of the streaming information. Section 5 discusses how this is exploited.

3.3 Problem statement

The problem of link prediction was modelled as follows:

Inputs:

- Directed graph $G = (V, E)$ where V are the Twitch users that have uploaded at least one stream and E the directed follow links between users.
- Streaming information S where S is a dictionary that maps from every user to a list of games they streamed in the past, and the time when the streams occurred.

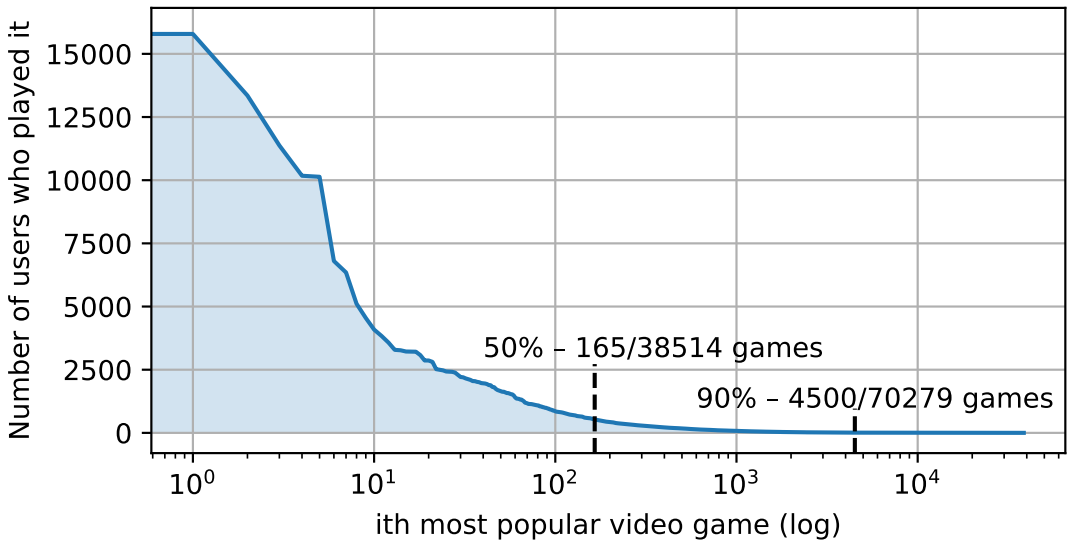

Figure 3.4: Distribution of videogames. A very small subset of games 165/38514 account for 50% of all the streams.

- Edge formation information T where T is a dictionary that maps from every edge in E to the time that edge was formed.
- Observable time interval: **train interval** $[t_{train}, t'_{train})$
- Time interval to predict the formation of new edges: **test interval** $[t_{test}, t'_{test}]$

Let $G[t,t']$ be the subgraph of G where only edges formed in the closed interval $[t,t']$ are present. Similarly, $S[t,t']$ corresponds to the streaming information corresponding to streams uploaded in the time interval $[t,t']$ and $E[t,t']$ to the edges created during the interval $[t,t']$. For the experiments in this work, t'_{train} and t_{test} are the same date, and correspond to the first query date (May 2018) when the streaming information was obtained.

The rapid and constant creation of new nodes in social networks is a limitation for link prediction algorithms. Thus, I ignored vertices created after t'_{train} and edges whose any of their endpoints is one of these ignored vertices. This is a common practice in the evaluation of link prediction algorithms [33].

Output: Ranked list of new edges in decreasing order of confidence L. The edges in L are predicted to appear during the test interval. L shouldn't contain any edge already present in the train interval. L could be interpreted as a map from every pair of vertices $(v_i, v_j) \notin E[t_{train}, t'_{train}]$ (E_{train} for future reference) to an assigned similarity score.

3.4 Evaluation

The evaluation of link prediction algorithms is not a trivial problem. There has been work addressing the specific problem of evaluation in link prediction [56] that pins down the reason for this difficulty to two main problems: **extreme class imbalance**

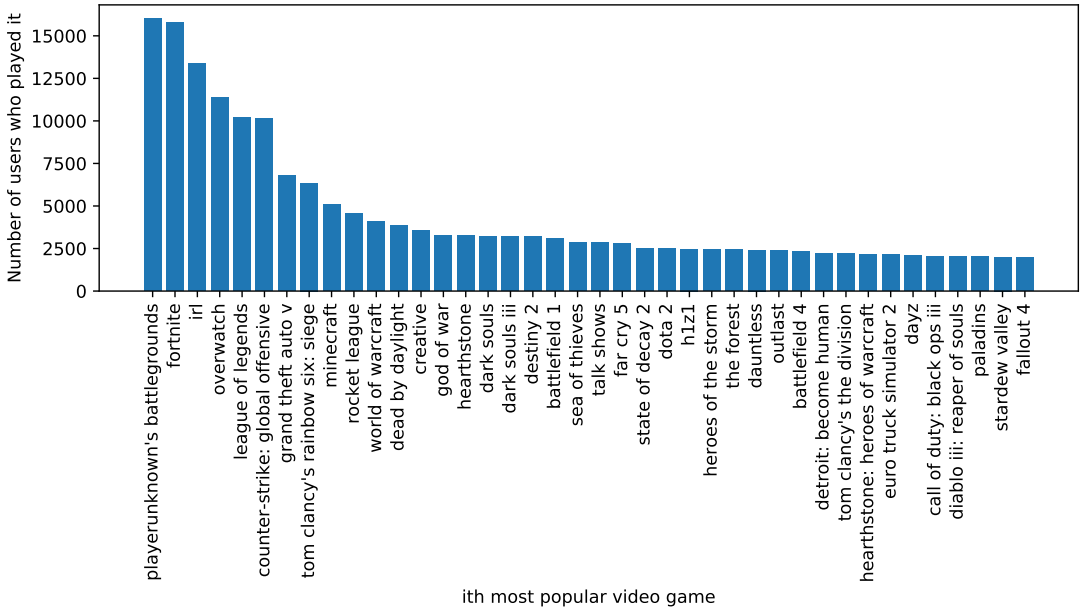

Figure 3.5: Popularity of the 40 most common video games. A small subset of games are very popular while the majority of games are not.

and **large size of graphs**. The first problem arises because true edges are a very small fraction of all possible edges, i.e. every possible binary combination of nodes. The second problem arises because link prediction algorithms are usually evaluated on social networks, where the number of nodes tends to be very high.

A common way to evaluate link prediction algorithms [33] is to calculate the intersection between the top n edges produced in the resulting ranked list and $E_{test} = E[t_{test}, t'_{test}]$ and divide it by n. This method is called a *fixed–threshold metric*. A common way to choose n is $n = |E_{test}|$. This metric could be understood as the probability that the classifier predicts an edge that is present in E_{test}. However, this metric is only a measure of the precision of the classifier. It is interesting to explore how the predictor performs for difference values of n. For example, we might be interested in predicting a very small set of edges $<< |E_{test}|$ with a high confidence.

3.4.1 Addressing class imbalance

Evaluating binary classifiers at different threshold levels (n) for imbalanced datasets is a common problem. A widely used and successful evaluation strategy is the use of Receiver operating characteristics (ROC) curves [18]. Especially, the area under the ROC curve (ROC AUC). Because ROC curves only consider the False Positive Rate and the True Positive Rate, they are agnostic to class imbalance. This can be very desirable, as the performance of a link prediction model evaluated via a ROC curve will be the same regardless of the class balance of the test set. The evaluation can be considered as more robust.

However, the use of ROC curves to evaluate link prediction problems is controversial [56]. Through multiple experiments on existing datasets and edge prediction algo-

rithms, Yang et al. 2015 [56] show that ROC curves can be deceptive to convey the actual performance of these algorithms. They argue that even though class-imbalance agnosticity creates more robust classifiers, they can be too optimistic about their performance in real scenarios.

To combat this problem, Yang et al. 2015 [56] suggest the use of Precision Recall (PR) curves –and area under the PR curve– instead [11]. PR curves are not agnostic to class imbalance, so they allow for more interpretable results that speak more closely to how the classifier would perform in practice.

Nevertheless, maximizing the area under the ROC curve will maximize the number of TP and minimize the number of FP, which intuitively is what makes a good classifier in the link prediction domain. For this reason, and because it's not practical to have an evaluation metric that changes based on the dataset or sample tested, I used area under the ROC curve to evaluate different choice of classifiers and parameters.

3.4.2 Addressing large graphs

Depending on the type of link prediction algorithm used, calculating L for every pair of possible vertices can be very expensive. Especially when the number of vertices is high.

$$|L| = \binom{|V|}{2} - |E_{train}| \tag{3.1}$$

The Twitch graph being used contains 53,201 vertices. E_{train} contains 572,341 edges. So using the equation 3.1, $|L| = 1,414,574,259$. It's not feasible to use such a large amount of testing points, so random sampling was used as an alternative[56].

An issue with random sampling is that the ratio of actual vs possible edges is very small. To be precise, the ratio of edges to non-edges is: $|E_{test}| : |L| = 295,392 : 1,414,574,259 \approx 1 : 4,789$. This means that in order to get a significant amount of edges from the minority positive class, a lot of edges from the majority negative class would have to be sampled.

A common practice to address this problem is to subsample the majority class until perfect balance. As shown by [56], this subsampling won't have any impact on the results when the classifier is evaluated using ROC curves. This is true because of ROC curve's agnosticism to class imbalance. Balancing the classes is also beneficial in order to train classifiers effectively. Most binary classifiers require a balanced class distribution to avoid the modelling of unwanted biases.

For that reason, in order to evaluate the experiments carried out, I used a randomized subsampling approach. For each experiment multiple sets of sample edges (*target edges*) were used. In order to obtain each set of target edges I sampled the same number of edges from $E[t_{train}, t'_{train}]$ and from $VxV \notin E[t_{train}, t'_{train}]$. The average result was used. The maximum positive/negative differences was used as the error.

Chapter 4

Exploiting topological information

This section addresses the choice of topological features. The goal is to obtain features from the topological data that can be used by a binary classifier to predict whether a link between two users exists or not. Throughout this section, the term *positive edge* refers to edges that are in the test interval. Conversely, the term *negative edge* refers to an ordered pairs of nodes that is not in the train or test interval. This section is a qualitative exploration of the usefulness of the different topological features for link prediction. The exact numerical impact that the features have is discussed in section 6.

Section 2.3 introduced a set of features for the undirected setting. However, the dataset at hand represents a directed graph, and some of the metrics had to be slightly adapted based on the meaning that directedness carries in Twitch. For the directed context, let $\Gamma(x) = \Gamma_{out}(x) \cup \Gamma_{in}(x)$. For nodes x and y, $score(x,y)$ represents the likelihood that an edge from x to y exists, i.e. $score(x,y) \neq score(y,x)$. x will be referred as the *follower* and y as the *followee*.

Predicting a link in a directed graph is not only a question of similarity, as similarity is a symmetric undirected metric. For that reason, the directedness of the edges has to be taken into account when creating useful topological features. Considering the undirected version of topological features is also interesting as similarity, albeit not exclusively, also carries useful meaning. Using the same classification as in section 2.3, the topological features used are introduced in the following sections. In order to evaluate the distribution of negative and positive edges for different features, a balanced sample of $100,000$ edges was created.

4.1 Neighbourhood based metrics

Common Neighbours (CN): Both the directed and undirected versions of the *CN* between two nodes were used as features. The undirected version is according to equation 2.7.

If we want to predict who x is going to follow in the future, one could argue that $\Gamma_{out}(x)$ gives us more insightful information than $\Gamma_{in}(x)$. I.e. knowing who x followed in the past will indicate who x is likely to follow in the future. Following that reasoning, if

many users in $\Gamma_{out}(x)$ follow another user y then it's likely that x will follow them too in the future. According to this reasoning a directed version of CN was introduced:

$$CN_{directed}(x,y) = |\Gamma_{out}(x) \cap \Gamma_{in}(y)| \tag{4.1}$$

The distribution of positive and negative edges for different values of CN for the directed and undirected versions can be seen in figure 4.1 and figure 4.2. Both show that positive edges have a larger number of common neighbours in average. The directed version shows a better separation of the negative and positive edges for $CN = 0$. This suggests that the directed version of CN is a more useful feature for link prediction.

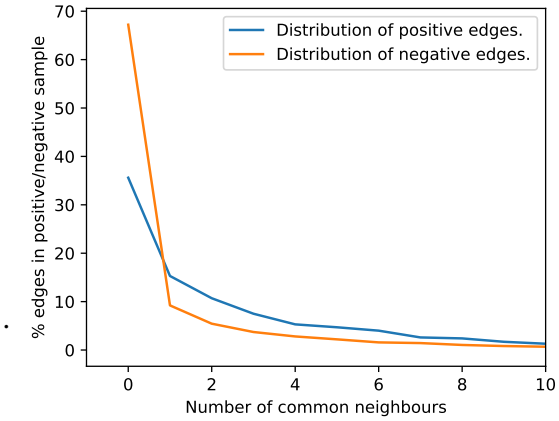

Figure 4.1: Distribution of positive and negative edges for different Common Neighbours (directed). Positive edges show a larger number of common neighbours.

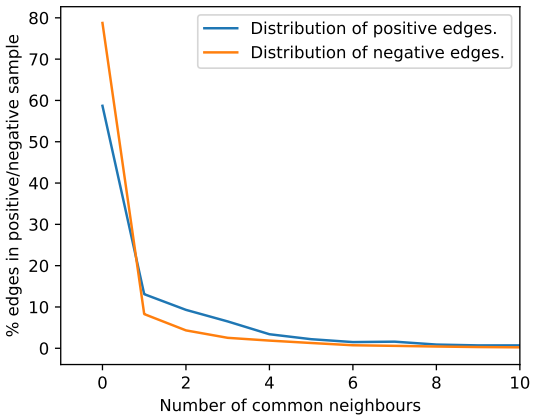

Figure 4.2: Distribution of positive and negative edges for different Common Neighbours (undirected). Positive edges show a larger number of common neighbours.

Jaccard's Coefficient (JC): Using the same reasoning to define $CN_{directed}$, a directed version of the JC can be defined as:

$$JC_{directed}(x,y) = \frac{|\Gamma_{out}(x) \cap \Gamma_{in}(y)|}{|\Gamma_{out}(x) \cup \Gamma_{in}(y)|} \tag{4.2}$$

However, for JC, the undirected and directed version of the feature displayed an almost identical distribution. Consequently, only the undirected version was displayed (figure 4.3). It seems that the only significant difference between the negative and positive edge distribution is the aggregated density when $JC = 0$ (no neighbourhood overlap). As expected, a considerably larger amount of negative edges have $JC = 0$. This was taken out from the graph to the legend for ease of visualization. A histogram with a variable number of bins was used to generate the graph (as JC's values are continuous). The aggregated density represents the relative size of the bars that were used to generate the curve. The absolute value itself is meaningless but allows a fair comparison of the trend between negative and positive edges.

Adamic-Adar Coefficient (AA): The distribution of positive and negative edges for different AA Coefficients is shown in figure 4.4. Again, $AA = 0$ was taken to the legend

and the aggregated density was used. Negative and positive edges show a different distribution and positive edges have a generally higher AA.

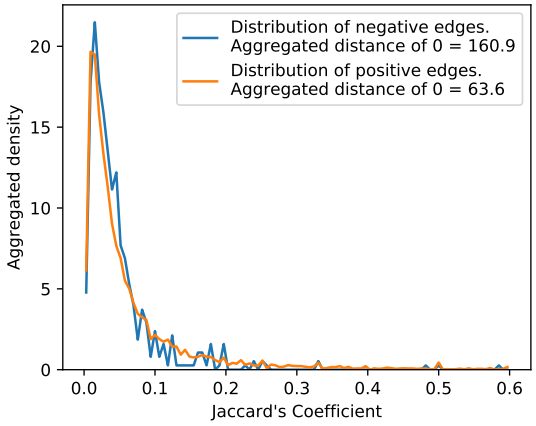

Figure 4.3: Distribution of positive and negative edges for different Jaccard's coefficients (undirected). More negative edges have a neighbourhood overlap of 0.

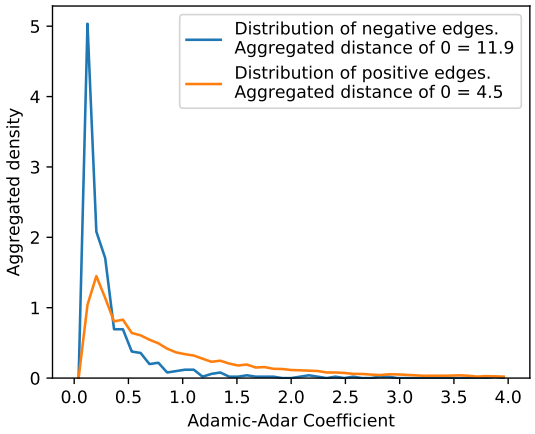

Figure 4.4: Distribution of positive and negative edges for different Adamic-Adar coefficients (undirected). Positive edges show a larger AA coefficient.

4.2 Distance based metrics

Shortest Path (SP): Both the directed and undirected versions of the shortest path metric were explored. The undirected version allows paths to go from vertex x to vertex y if an edge exists between them, regardless of its direction. For the directed version, $SP(x,y)$ only allows a path to form along the direction of the edge, starting in x and finishing in y. Figure 4.5 and figure 4.6 show the distribution of positive and negative edges for different lengths of SP. Positive edges have a shorter shortest path between them. More positive than negative edges lack a path between them. Figure 4.6 shows that very few positive edges have a 1-shortest path connecting them. This means that follows are hardly ever reciprocal.

Rooted PageRank (RPR): The undirected version of PageRank can be used as a measure of distance (and consequently similarity) in the network, as the directionality of the edges is not taken into account. For the directed version, $RPR(x,y)$ could be thought as an extension of the argument given for $CN_{directed}$. x is interested in $\Gamma_{out}(x)$ and for each $z \in \Gamma_{out}(x)$, the same argument applies until the interest chain reaches y.

Figure 4.7 and figure 4.8 show the distribution of positive and negative edges for different values of RPR. Positive edges have a higher PageRank value than negative ones. The directed version seems to do a better job at separating positive and negative edges.

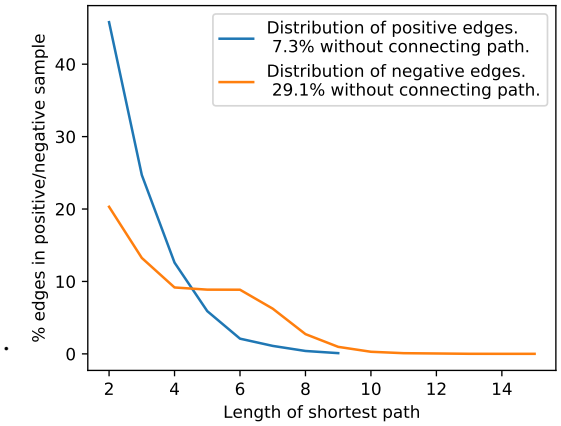

Figure 4.5: Length of shortest path for positive and negative edges of the test set. Positive edges have a shorter shortest path between them. More positive than negative edges lack a path between them.

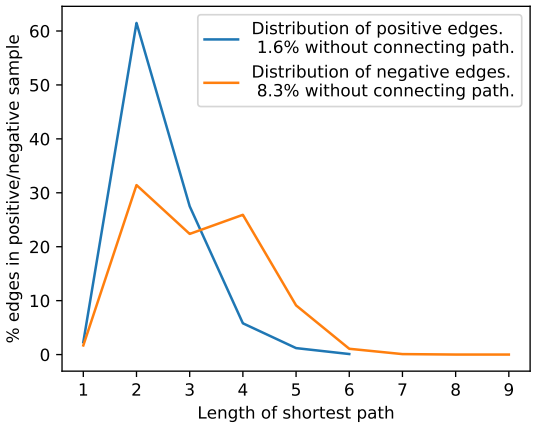

Figure 4.6: Length of shortest path for positive and negative edges of the test set (undirected version). Positive edges have a shorter shortest path between them. More positive than negative edges lack a path between them. Reciprocal follows are very uncommon.

4.3 Social theory based metrics

Centrality: Three metrics were explored to capture centrality: **In-Degree (InD), PageRank (PR) and Betweenness Centrality (BC)**. *InD* is the most naïve version of centrality, and was chosen over Out-Degree to capture the notion of incoming weight of importance. In the context of centrality, directed *PR* determines the importance of a node according to the incoming weight that that node receives. The undirected version of *PR* determines the importance of a node by how close it is from all the other nodes in the network. For *BC*, the undirected version is more appropriate as we just want to capture the density of shortest paths through the network.

InD, *PR* and and *BC* are node-wise scores. As mentioned in the beginning of the chapter, only the centrality of the followee is examined, i.e. $score(x,y) = InD(y)$ or $PR(y)$ or $BC(y)$. Figure 4.9, figure 4.10 and figure 4.11 show the distribution of positive and negative edges using *InD*, *BC* and *PR* respectively. The distribution is very similar for the three metrics. Negative edges show a very pronounced spike for smaller values while positive edges show a smaller small-value spike and a more uniform distribution.

Clustering: The **Local Clustering Coefficient (CC)**, as proposed by Newman et al. 2001 [41] was used. Figure 4.12 shows the distribution of positive and negative edges. There is no obvious distinction between positive and negative edges. If anything, the negative edges show a slightly higher *CC*. This might be due to *CC* being defined on the basis of $\Gamma_{out}(x)$ and the fact that streamers don't usually follow many users –but rather are followed by many.

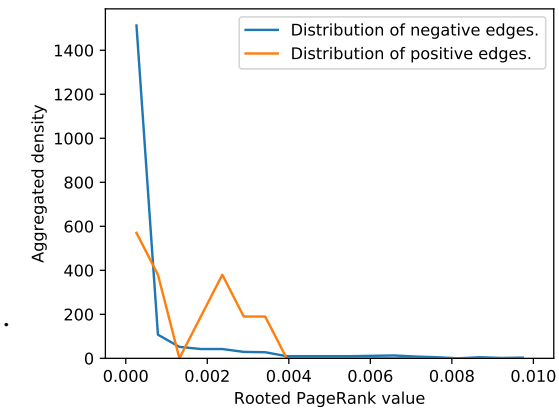

Figure 4.7: Rooted PageRank for positive and negative edges of the test set (directed). Positive edges have a higher PageRank value than negative ones.

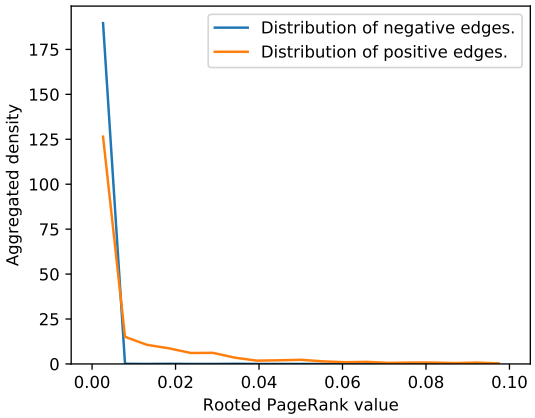

Figure 4.8: Rooted PageRank for positive and negative edges of the test set (undirected). Positive edges have a slightly higher PageRank value than negative ones.

4.4 Summary and conclusions

The following conclusions can be drawn from the qualitative inspection of the distribution of negative and positive edges for different topological features:

- Incorporating directedness into the features generally results in a better ability to separate negative and positive edges. This is particularly true for neighborhood overlap and rooted PageRank.
- Common Neighbours and Adamic-Adar Coefficient are good neighbourhood based metrics to separate negative and positive edges.
- Jaccard's Coefficient doesn't do a good job at separating negative and positive edges. Because *JC* only differs from *CN* in the normalization factor ($|\Gamma_{out}(x) \cup \Gamma_{in}(y)|$) it's very likely that the success of *CN* comes from the fact that followees are popular users and have a high number of followers ($\Gamma_{in}(y)$ is high).
- Both Shortest Path and Directed Rooted PageRank are good distance based metrics for separating negative and positive edges. Undirected Rooted PageRank doesn't do as well.
- Most of the metrics show a common pattern in the negative-positive edge distribution: negative edges have a pronounced spike in small values of the metric, while positive edges show a more uniformly spread distribution across higher values.
- Central nodes are more likely to be followed than non central nodes. In-Degree, PageRank and Betweenness Centrality are good metrics to capture centrality. The main reason why centrality is a good metric is because followers are usually very uncentral nodes (with hardly no incoming edges). The skewed distribution of centrality can be appreciated in Figure 3.2. Li et al. 2011 [32]'s claim holds

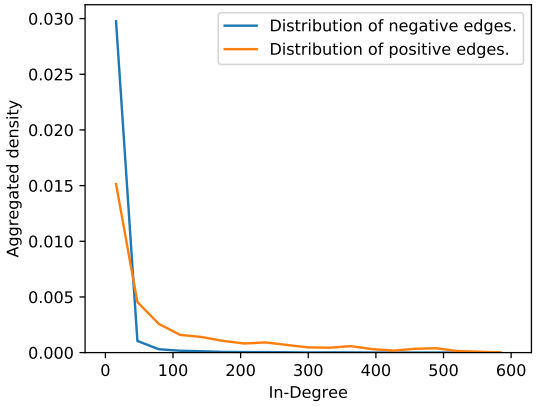

Figure 4.9: In-Degree for positive and negative edges of the test set. Positive edges have a higher In Degree value than negative ones.

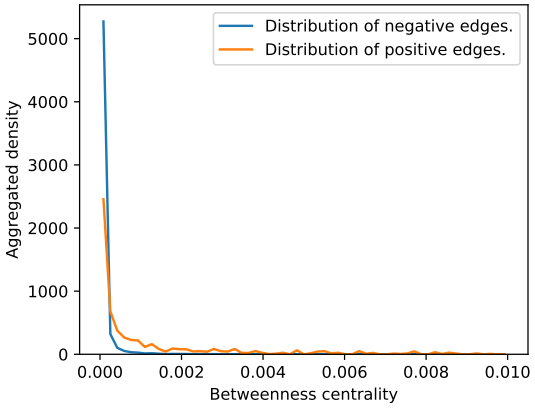

Figure 4.10: Betweenness centrality for positive and negative edges of the test set . Positive edges have a higher BC value than negative ones.

for Twitch.

- The clustering coefficient of a followee is not a good metric to separate negative and positive edges.
- Follow edges are hardly ever reciprocal. This shows that users don't only form links on the basis of similarity in Twitch, as similarity is symmetric.

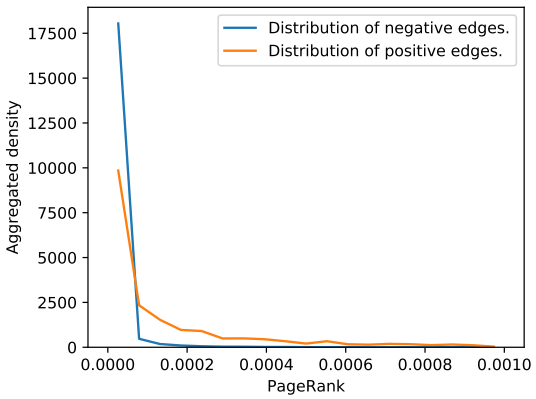

Figure 4.11: PageRank for positive and negative edges of the test set. Positive edges have a slightly higher PageRank value than negative ones.

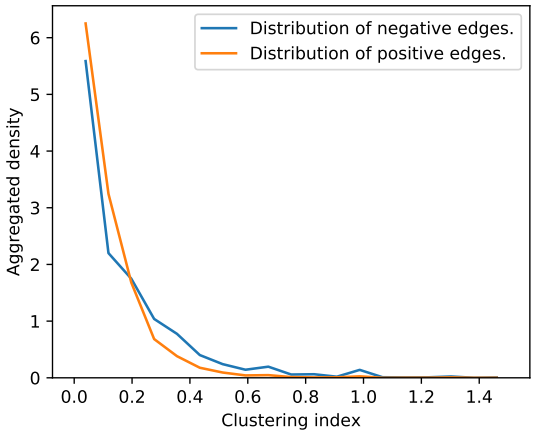

Figure 4.12: Local Clustering Coefficient for positive and negative edges of the test set (undirected version). No obvious distinction between positive and negative edges.

Chapter 5

Exploiting non-topological information

This chapter addresses the exploitation of non-topological information. The goal is to extract a set of features from the available non-topological data that can be used by a binary classifier to predict whether a link between two users exists or not.

Handling non-topological features is not a trivial task. In the context of Twitch, non-topological features are all node-specific. However, the goal is to create a set of features that characterize node pairs that form a link. This work is concerned with directed links, which limits the use of simple aggregation functions that are commutative like sum and multiplication.

Commutative aggregation functions are still useful because they can capture a notion of similarity, which is undoubtedly useful to predict the formation of a link. However, the use of non-commutative aggregations between features is also necessary if directionality wants to be captured.

A trivial non-commutative way to combine features is to include the original node features in the feature set in a fixed way. For example, given two nodes n_1, n_2 and their aggregated view counts vc_1, vc_2, the way to combine the aggregated view counts is to use vc_1 as the first feature for the binary classifier and vc_2 as the second one. This however puts the burden of finding the interrelations between features in the classifier, and thus more complex models will have to be used. This is not problematic when the original features can be represented by a single scalar, but grows in complexity when the original features are multi-dimensional vectors.

For that reason, simpler node-specific attributes were included for both nodes in the node pair. Vector–based attributes like country or streaming information were studied in finer detail.

5.1 Demographic features

For two users u_1 and u_2, the original demographic data was used to create a set of demographic features representing the link between them:

- **User is partner:** Two features are created. The first feature takes value of 1 if u_1 is a partner, and a value of 0 otherwise. Same logic for the second feature and u_2.
- **User is mature:** Two features are created. The first feature takes value of 1 if u_1 makes mature content, and a value of 0 otherwise. Same logic for the second feature and u_2.
- **Common country:** Single binary encoded feature representing whether u_1 and u_2 are from the same country.
- **Number of views:** Two features are created. The first feature encodes the number of aggregated views for u_1 and the second feature the number of aggregated views for u_2.

5.2 Streaming information

This section addresses the question of how streaming information can be exploited optimally. I decided to model the streaming information data as a matrix S where S is a (number–of–users x number–of–videogames) matrix. Each row of S is a binary multi-hot encoded vector such that $S_{i,j} = 1$ if user i ever streamed video game j and 0 otherwise.

The alternative of splitting the streaming information into temporal intervals was considered. However, this approach was rejected for two reasons:

- As shown in figure 3.3 most of the streaming information is only available during the last 60 days. This is too short of a time to capture evolution patterns in the streaming behaviour of a user.
- The goal of using the streaming information is to obtain the most up–to–date information of the user's streaming behaviour and get insight into latent preferences when it comes to gaming. Capturing the evolution of these latent preferences is likely not going to help in characterizing the streaming behaviour.

For these reasons, the streaming information is exclusively used to create a symmetric set of features between the users in an edge that tries to capture similarity in their streaming pattern. Consequently, in order to guide the feature selection, an undirected version of the graph was considered. The ability to predict whether two nodes are connected was used as the way to evaluate how useful the features are. Then, A similarity score between streaming patterns was produced and binarized for many thresholds using a ROC curve and the area under the curve was used to measure performance.

The streaming frequency of the followee was also used as a feature. This was motivated by the fact that active users are more likely to create an engaging community and cause other users to follow them [25]. As discussed in section 3.2, partners can save their videos for 60 days while regular users for 14. To bridge that mismatch I only used the last 14 days to create the frequency feature.

$$FREQUENCY_j = \text{number of individual streams by user j in the last 14 days}$$

5.2.1 Shared video games

There is some parallelism between the nature of the streaming information S and the BOW approach to text modelling. A very common dataset used in the evaluation of link prediction is the citations network, where users are connected if they have collaborated more than a set amount of time. Similarly to the streaming information, users in this dataset have a set of publications that they've written. Al Hasan et al. [8] found *keyword match count* to be a very useful feature to predict the presence of a link in citation networks.

In the video game streaming problem, the equivalent to the *keyword match count* would be the *game match count*, i.e. the number of video games two players have in common in their streaming history. This is also a very intuitive similarity metric. Users only stream a game when they are interested in it. Other users streaming that same game are therefore clearly users of interest.

The number of shared video games between two users i and j ($SHARED_{i,j}$) is calculated as the dot product of the vectors corresponding to i's and j's encoding:

$$SHARED_{i,j} = S_i \cdot S_j$$

5.2.2 Feature Embeddings

The thought that motivated the next iteration was the following: users have preferences when it comes to video games. For example, a user might like action games and adventure games but not be interested in strategy games. Or a player might prefer mainstream games to non-mainstream games. If we were able to identify the video game aspects that differentiate the users, we would be able to represent each user by their preferences on these aspects and use these preferences as features to obtain a similarity metric between them.

It's important to note that uncovering these preferences doesn't serve as a substitute or improvement on the *SHARED* feature as they serve different purposes. Besides, feature embeddings will help with the problem of string-based game titles. E.g. *Call Of Duty 2* and *Call of Duty II* are likely to be embedded similarly and thus bridge the gap between different game spellings.

In order to generate this preference-representation of the users, I used feature embeddings. The embedding takes S and produces a transformed $S_{transformed}$ where each user is represented with $k <$number-of-videogames s.t. $S_{transformed}$ is a ($|V|$xk) matrix.

Different embedding algorithms have their own evaluation metrics that they internally use to determine the quality of the embedding. However, the goal of generating these embeddings is to help the accurate prediction of links. For that reason I opted for an external evaluation of the embedding algorithms, i.e. how useful are the produced embeddings to predict whether or not a link exists between two users.

The resulting streaming embedding for a user i represents i's streaming preferences. It follows to assume that similar users should have similar embeddings. For that reason, given two users i and j, I used the negative inverse of the distance between their corresponding embeddings as their associated similarity score.

Choosing a distance function is not a trivial task. I explored both Euclidean and cosine similarity as they are standard similarity metrics used in the literature. The Euclidean similarity was obtained by taking the negative inverse of the Euclidean distance. However, I also explored the use of distance functions tailored to the method in question. The choice of these alternative distance functions will be discussed in the upcoming sections.

I explored different embedding algorithms. Namely, Truncated SVD, Principal Component Analysis, Non-negative Matrix Factorization, Doc2Vec and Latent Dirichlet Allocation. For each of these methods different parameters had to be explored to find the optimal performance. A common parameter to be explored for all the different embedding algorithms is the number of embedded dimensions, k. This value is dependent on the dataset in question so experimental trial and error is the only viable approach when the number of latent dimensions is unknown.

5.2.2.1 Truncated SVD and PCA

This subsection discusses the use of SVD and PCA to create a feature embedding of S. In order to solve the matrix decomposition, the eigenvalues and eigenvectors of the matrices AA^T and A^TA have to be obtained. This can be obtained through multiple numerical methods. A widely used numerical method to find eigenvalues and eigenvectors for large matrices is ARPACK [31]. Halko et al. 2009 [24] introduce a faster alternative that relies on a randomized iterative approach. Both algorithms were tested.

A streaming matrix S that uses the entire dataset would result in S having dimensions (53,201 x 70,279). This is a total of 3,738,913,079 elements. Most of these entries are zero as users don't usually play more than 100 different games, which allows the use of sparse matrices where only the nonzero values are represented. However, if the data has to be standardized before the factorization –which is the case for PCA– all elements have to be stored.

For that reason, I explored the impact of using smaller fractions of the dataset by randomly sampling the users. The performance of all methods was mostly unaffected for smaller fractions of the dataset (of around 50%). The fact that smaller fractions of the dataset didn't hinder the performance of the model allowed me to speed up the parameter search and the ability to fit the reduced streaming matrix in memory more easily. Nevertheless, the entirety of the dataset was used for the final experiments. Another factor that allowed this was the reduced number of video games being considered, whose impact will be discussed below.

Both ARPACK and a randomized solver were tested but both resulted in a similar performance. Both cosine and euclidean similarity were tested but both resulted in a similar performance. The results can be seen in table 5.2.

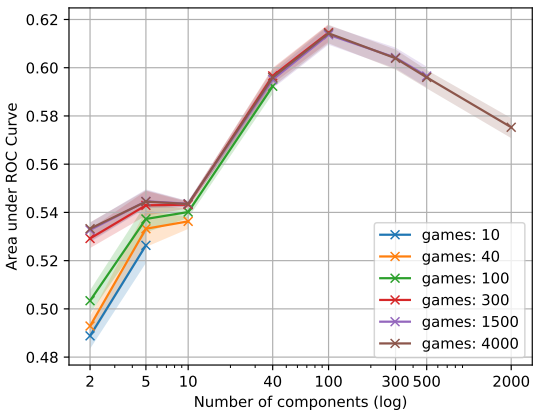

Figure 5.1: Performance of Truncated SVD embedding and euclidean similarity for link prediction. Different portion of the top most popular games used. Using euclidean similarity, randomized solver and a 40% of the dataset. 100 components resulted in the best performance regardless of the number of games used.

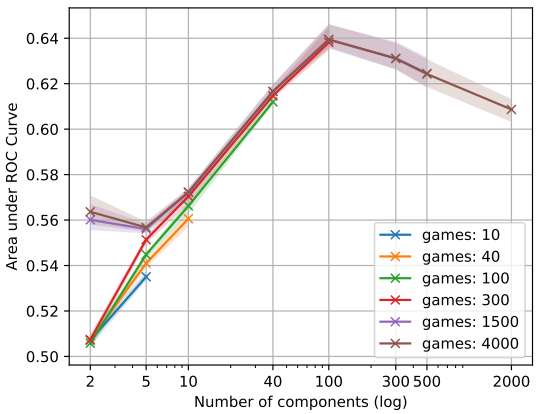

Figure 5.2: Performance of PCA embedding and euclidean similarity for link prediction. Different portion of the top most popular games used. Using euclidean similarity, randomized solver and a 40% of the dataset. 100 components resulted in the best performance regardless of the number of games used.

The results of the experiments for SVD and PCA are shown in figure 5.1 and figure 5.2 respectively. The specific values are shown in table 5.2. Both SVD and PCA show a very similar performance.

They both peak at 100 components. Interestingly, this consistently happens for 300,1500 and 4000 games. As shown in figure 3.4, the percentage of users covered using 300 and 4000 games is quite different (58% vs 89%). However, 300 seems to be enough to capture the underlying streaming preferences, at least to the extent PCA and SVD are able to.

Nevertheless PCA seems to perform slightly better than SVD. This was expected, as contrary to SVD, PCA standardizes S prior to factorizing it. This allows each original dimension (video game) to be considered equally without allowing the different mean or variance to weight it more or less. As shown in figure figure 3.4, the popularity of different games varies dramatically.

5.2.2.2 Non-negative Matrix factorization

This section discusses the use of NMF to create a feature embedding of S. The streaming content at hand has a lot of similarities with the problem of recommender systems where there are users and items and the rating-based interactions between users and the items is recorded.

The main difference is that in this context, ratings are binary, i.e. if a player likes a game they will stream it, and if they dislike it they won't. This however has the limitation of missing data. Players might very not know about a video game, and that

doesn't mean they didn't like it.

I explored the use of two solvers based on two SVD processes each. The first solver, *NNDSVD* exploited sparse factorization, which is better at creating *parts-based representation*. The second solver, NNDSVDa doesn't impose sparsity constraints. L1 and L2 regularization was also explored in order to avoid overfitting and improve generalization.

NNDSVD showed a better performance, so I defaulted all my experiments to use *NNDSVD*. This is not only beneficial because of the increase in performance. Sparse representations are easier to interpret because of their pseudo-binary quality. The results can be seen in table 5.2. The same concern about the time and space complexity of PCA applies in NMF. I also used a reduced portion of the dataset and explored the use of a limited number of video games.

In contrast to SVD and PCA, the distance metric used to find the similarity between two embeddings affected the performance drastically. Figure 5.3, figure 5.4, figure 5.6 and figure 5.5 show the performance for each distance metric. Cosine similarity, NVSC1 and NVSC2 improve performance significantly over euclidean similarity. NVSC marginally improve cosine's similarity average performance but they are much more noisy. This hints that cosine similarity is a more robust metric. Because the increase in performance was so marginal and the increase in noise so dramatic, I decided to stick with cosine similarity to explore the impact of regularization. The specific values can be seen in table 5.2.

The results of applying regularization can be seen in figure 5.7 and figure 5.8. Both L1 and L2 regularization were explored but only L2 seemed to improve the results ($\rho = 0$). Other values of α were explored (as shown in table 5.2) but 0.5 resulted in the best performance. It's also interesting to observe that the performance of NMF for a small number of components (2 and 5) gets worse the more games we use. This was not the case when regularization wasn't used, as shown in figure 5.4. A possible explanation is that the regularization cost outweighs the optimization cost when the mismatch the W's dimensions is very high.

Another thing to notice is that the optimal number of components changed to 300 when regularization was used. A higher number of components might hint a higher granularity in the quality of the components.

5.2.2.3 Latent Dirichlet Allocation

This subsection discusses the use of LDA to create a feature embedding of S. I explored the effect of using different numbers of games, different similarity metrics (euclidean and cosine similarity), and different encodings of S (standard vs TF-IDF).

Applying TF-IDF in the context of video game streaming information can be very desirable because of the very skewed distribution of games, similar to the distribution of word frequencies in natural language, as shown in figure 3.4. That way, rarer games

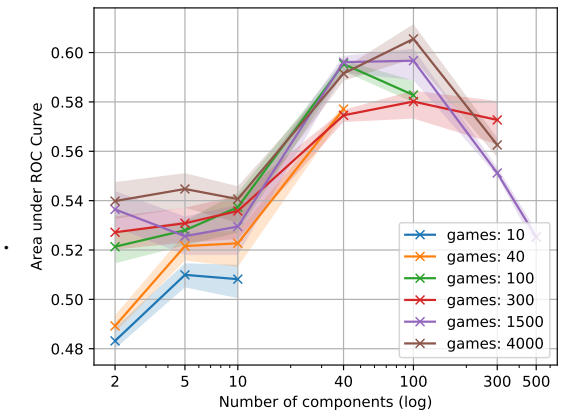

Figure 5.3: Euclidean similarity

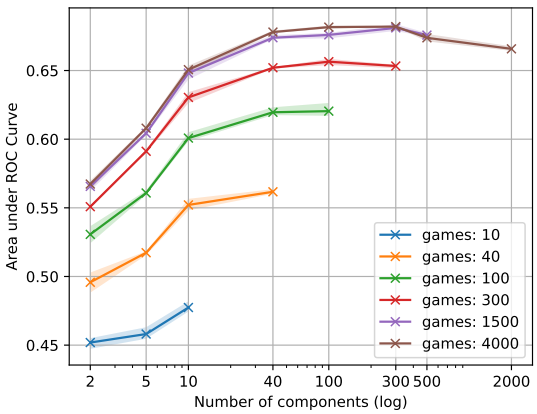

Figure 5.4: Cosine similarity

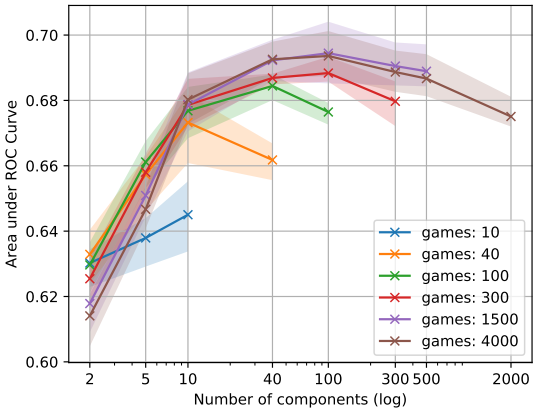

Figure 5.5: NVSC1

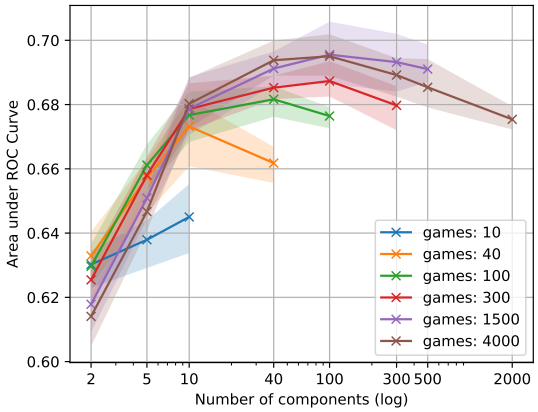

Figure 5.6: NVSC2

Link prediction using NMF embeddings for different distance metrics. 100 components result in the best performance accross all distance metrics. NVSCs and cosine similarity perform significantly better than euclidean similarity.

will be given the same weight as more common ones.

$$T_{i,j} = S_{i,j} * log(\frac{N}{df_j}) \tag{5.1}$$

TF-IDF is usually defined in the context of text and documents. For this context, the TF-IDF version of the binary value encodes whether a user i streamed game j ($T_{i,j}$) is given by equation 5.1. S is the streaming matrix, N the number of users and df_i the number of users that streamed video game j.

The probabilistic nature of LDA lends itself to online implementations using SGD-based methods. This is beneficial for large datasets, as it's not necessary to hold the entire dataset in memory as NMF as other factorization methods require. LDA also results in faster convergence partly due to the need of less iterations over the data. For that reason I was able to use the entirety of the dataset.

The results of the experiments can be seen in figure 5.9, figure 5.10, figure 5.11 and fig-

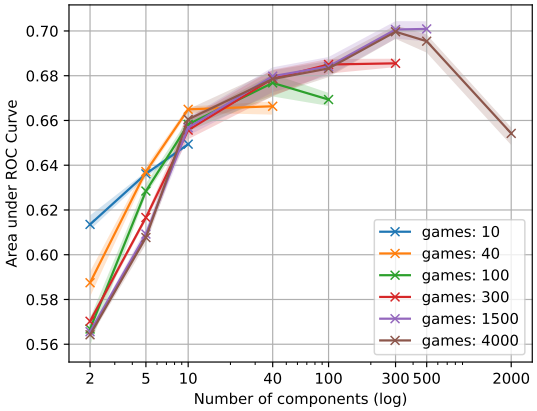

Figure 5.7: Link prediction using NMF embeddings, cosine similarity and L2 regularization with α=0.5. 300 components result in the best performance.

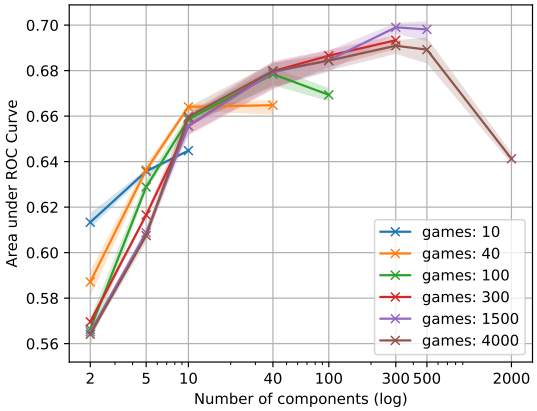

Figure 5.8: Link prediction using NMF embeddings, cosine similarity and L2 regularization with α=1. 300 components result in the best performance.

ure 5.12. Overall, LDA performed considerably worse than SVD, PCA and NMF. For comparison see table 5.2. Encoding S using TF-IDF resulted in a dramatic improvement in performance. When the standard encoding was used, the performance didn't even surpass random (0.5 AUROC). Both euclidean and cosine similarity resulted in very similar performances.

Something unexpected is the fact that the less number of games used, the better the performance is. This is especially pronounced when the standard encoding was used, which might be an indicator that using a BOW encoding poses many problems, arising from the skewed game distribution mentioned before. However, although to a much less extent, using 40, 100 and 300 games still performs better than using 600 and 1000 even when TF-IDF was used. This, together with the poor overall performance suggests that LDA might not be an optimal algorithm to embed the streaming information S.

5.2.2.4 Doc2Vec

This section discusses the use of *Doc2Vec* to create a feature embedding of S. As mentioned in section 2.4.1.4, the context window is variable. Because in the streaming context we are dealing with unordered words (vector of games), I used the BOW version of doc2vec (PV-DBOW).

Despite the high amount of free parameters Doc2Vec uses, the parameter updates during the training process are sparse, as only those words that appear in the documents trigger updates, so the process is very efficient [30]. This is a clear advantage over the embedding methods used before, as this allowed the use of the full-size S without reducing the number of games or the portion of the dataset used.

However, the choice of number of components and the ratio at which popular words (games) are subsampled still needs to be explored as it is domain dependent. Le et al. 2014 [30] suggest a value of 0.75 for the NS exponent. However, Caselles-Dupré et al.

2018[13] explored the use of Doc2Vec for recommendation systems and showed that a value of -0.5 delivered the best performance. A value of -0.5 results in rare words being sampled more often than common words.

I explored the performance of predicting links using Doc2Vec embeddings for different embedding sizes and different NS exponents as explained before. I tried euclidean and cosine similarity as a similarity metric between embeddings.

Figure 5.13 shows the results of the experiments. The exact values can be seen in table 5.2. The overall performance was significantly worse than NMF, SVD and PCA. A similar performance to LDA was obtained.

Caselles et al. 2018 [13] 's recommended NS exponent value (–0.5) resulted in the worst performance, and the original NS value proposed by Le et al. 2014 [30] (0.75) resulted in the best performance. This shows that sampling rare games more frequently is not beneficial to create a better embedding. The choice of similarity function didn't have an impact in the performance either.

5.3 Evaluation and interpretation

Throughout this chapter a set of non-topological features based on demographic information was proposed. Besides this, different ways of exploiting the streaming information were studied. Table 5.2 shows the summarized performance of the different embedding algorithms and parameters.

5.3.1 Visual inspection of non-topological features

The ability of the non-topological features to separate negative and positive edges was visually inspected in the same manner as topological features were inspected in section 4.

Figure 5.14 and figure 5.15 show the distribution of positive and negative edges for aggregated views of the follower and the followee respectively. The followee has a higher number of aggregated views for positive edges. However, the follower shows a similar number of views for both positive and negative edges.

Figure 5.16 shows the distribution of positive and negative edges for the followee's streaming frequency over the last 14 days. A larger number of followees in negative edges have a frequency close to zero. For positive edges, the followee streamed with a slightly higher frequency than in negative edges.

Figure 5.17 shows the distribution of positive and negative edges for the shared number of games. Users in positive edges share a larger number of games. 1 shared game is a clear breaking point between negative and positive edges.

Figure 5.18 shows the distribution of positive and negative edges for different values of NMF cosine similarity. Positive edges have a generally higher cosine similarity than negative edges. Negative edges show a pronounced peak for NMF=0 that corresponds to the case where no NMF feature has a shared non-zero value between users.

Feature	Positive edges (%)	Negative edges (%)
Is partner (follower)	89.0	11.0
Is partner (followee)	88.3	11.7
Is mature (follower)	55.3	44.7
Is mature (followee)	52.6	47.4
Common country	89.0	11.0

Table 5.1: Edge distribution for different non–topological features. High difference between positive and negative edges mean the feature is able to separate the negative from the positive edges.

Table 5.1 shows the edge distribution for the features: is-follower-partner, is-followee-partner, is-follower-mature, is-followee-mature, and common country. The is-partner features seem to separate positive and negative edges very effectively, probably due to partner users being more active. The is-mature features are not able to separate negative and positive edges well. The common country is able to separate negative and positive edges very well. Users that are from the same country are more likely to share a link between them, which makes sense as they not only share language but geographical location.

5.3.2 Exploiting individual NMF-embedded features

Due to NMF's clear dominant performance, NMF's factorization was used as the de-facto method to obtain the feature embeddings of S. The settings that resulted in the best performance were used: **NMFsolver = NNDSVD, α=0.5, ρ=0** (table 5.2). Taking a closer look at the resulting embedded dimensions can provide insight in how to best use them to generate features for the link prediction classifier.

An advantage of NMF's factorization is that the resulting features can be interpreted thanks to how the factorization is set up. In a collaborative filtering approach where the rows of X correspond to users and the columns correspond to the items users interact with, W and H can are interpretable.

- The columns of W represent the basis of the new latent representation.
- The columns of H embed the games in the same low-dimensional space as the users were embedded.
- The rows of H correspond to the component weights. That is, each value in H's row represents how to sum the original dimensions to create the basis vector. This can be understood as a weighting of the original dimension.

So using the rows of H, each new component can be represented as a weighted compound of the original dimensions. In the case of games being the original dimensions, looking at this weighting can draw insight into what each of the new embedding dimensions represents. Table 5.3 shows the top associated games for a sample of dimensions. For each dimension, the games with the five highest associated values are detailed. A comment detailing the relationship that the games have in common is also provided.

Embedding method	Similarity metric	components	AUROC
SVD solver = ARPACK	euclidean	100	0.615 ±0.004
SVD solver = randomized	euclidean	100	0.620 ±0.009
SVD solver = randomized	cosine	100	0.616 ±0.006
PCA solver = ARPACK	euclidean	100	0.631 pm 0.012
PCA solver = randomized	euclidean	100	0.642 pm 0.006
PCA solver = randomized	cosine	100	0.638 pm 0.007
NMF solver = NNDSVDa, α=0, ρ=0	cosine	300	0.663 ±0.005
NMF solver = NNDSVD, α=0, ρ=0	cosine	300	0.682 ±0.004
NMF solver = NNDSVD, α=0,ρ=0	euclidean	100	0.606 ±0.008
NMF solver = NNDSVD, α=0, ρ=0	NVSC1	100	0.692 ±0.010
NMF solver = NNDSVD, α=0, ρ=0	NVSC2	100	0.693 ±0.011
NMF solver = NNDSVD, α=0.25, ρ=0	cosine	300	0.699 ±0.005
NMF solver = NNDSVD, α=0.5, ρ=0	cosine	300	**0.701 ±0.004**
NMF solver = NNDSVD, α=1, ρ=0	cosine	300	0.699 ±0.003
NMF solver = NNDSVD, α=1, ρ=0	cosine	300	0.699 ±0.003
NMF solver = NNDSVD, α=1, ρ=0.5,	cosine	100	0.687 0.004
LDA encoding = standard	euclidean	60	0.482 ±0.007
LDA encoding = TFIDF	euclidean	30	0.559 0.005
LDA encoding = standard	cosine	20	0.481 0.002
LDA encoding = TFIDF	cosine	30	0.559 0.008
Doc2vec	cosine	0.75 (ns exp)	0.593 ±0.009
Doc2vec	euclidean	0.75 (ns exp)	0.592 ±0.007

Table 5.2: Different embedding methods and their performance in link prediction. The components cell refers to the number of components that resulted in the best performance

Component no.	Games	Notes
5	'dark souls ii: scholar of the first sin', 'bloodborne: the old hunters', 'nioh', 'dark souls iii', 'dark souls ii'	Dark soul games and similar action dark-themed
252	'totally accurate battle simulator', 'infestation: newz', 'the forest', 'PlayerUnknown's Battlegrounds', 'battalion 1944'	Battle action games
1	'fornite', 'roblox', 'fortnite', 'god of war', 'h1z1'	Online multiplayer battle games
26	'kerbal space program', 'terraria', 'minecraft', 'roblox', 'factorio'	Pixel based construction/crafting games for a younger audience
281	'super mario bros. 2', 'maldita castilla', 'refunct', 'super mario world', "chip 'n dale: rescue rangers"	arcade old–school games
105	'call of duty: black ops iii', 'call of duty: infinite warfare', 'call of duty: wwii', 'call of duty: black ops ii', 'call of duty®: wwii'	Call of duty franchise
187	"tom clancy's rainbow six: siege", 'fortnite battle royale', "playerunknown's battlegrounds", "tom clancy's rainbow 6: patriots", "tom clancy's rainbow six siege"	genre: shooters
221	'rise of the tomb raider', 'far cry: primal', 'tomb raider', "uncharted 4: a thief's end", 'just cause 3'	Genre: Action-adventure games
7	'mega man x', 'mega man x3', 'mega man x2', 'mega man', 'mega man 2'	Mega man franchise

Table 5.3: Example of top games for different components using the best-performing NMF parameters. Games in individual components share a theme.

It's also worth noting that the NMF factorization is able to overcome the string-based game encoding limitations. For example, component no. 1 combines 'fornite' and 'fortnite' into a single component.

It can be observed that different features correspond to clear subgenres, video game franchises or other game subtypes. This is very interesting, as the factorization was able to capture meaningful underlying user preferences. These preferences can be easily understood and have a real potential to define similarities between users.

A low dimensional embedding of the games allows to capture similarity between games for applications like game recommendation or game clustering. Game recommendations are outside the scope of this project, however, it's interesting to observe how similar games are close in this embedding. Figure 5.19 shows a 2D embedding using the original NMF factorization and PCA of the top 100 games. This is a limited visualization because the dimenions were reduced from 300 to 2. However, it's still possible to notice similar games being close together. For example, *call of duty: black ops ii* and *tom clancy's rainbow six: siege* are both similar first person shooter games, and *creative* is a sandbox game created inside the game *fortnite*.

The similarity metric obtained by using the cosine distance between the NMF-based embeddings for the two users is used as a feature for link prediction. However, I believe the feature embedding produced by the NMF factorization can be used further.

As discussed in section 5.2.2.2, *NNSVD* was used as a solver for NMF. This results in sparse factorizations that, according to Hoer et al. 2004 [26] are better at creating parts–based factorizations. Figure 5.20 shows a heatmap–based representation of the values of the components for a sample of users. It can be seen that only a small subset of components for each user are non-zero. This makes sense if the meaning of the features, as shown in table 5.3, is considered. Users usually like a limited number of genres of games, and this set of features captures the nature of that sparse user preference.

A link predictor will definitely benefit from knowing the overall similarity of the user preferences (the embedded features). However, I believe using the components for each of the embedded features individually can also be very helpful. It could be the case that some components are more important than others at suggesting similarity. For example, two users having non-zero values in component no. 34 in table 5.3 (arcade old school inspired games) might have a higher probability of a link existing between them than if they both had non-zero values in component no. 187 (genre: shooters). This is the case because component no. 34 encodes a more niche genre that less users play.

For that reason, inspired by how Al Hasan et al. 2006 [8] handled aggregate features, I decided to add the pair-wise product between the embedding components of both users as individual features. This pair-wise product has non-zero values for those components that both users had non-zero values on. It also has higher values if both users agree on that component.

The following conclusions can be drawn from the exploration of non-topological features throughout this chapter:

- Non-Negative Matrix factorization is the most optimal method to embed the available streaming information. The use of cosine similarity as a similarity metric is essential. Truncated SVD and PCA produce decent embeddings but are not as good as NMF. LDA and Doc2Vec are not able to produce quality embeddings. Besides, sampling rare video games more often doesn't improve results which hints that contrary to in the text domain, rare games don't carry more meaning.
- The aggregated views of the follower is not able to separate negative and positive edges. However, the aggregated views of the followee is. This makes sense, as the more popular a user is the more likely other people are to follow them.
- Followees in negative edges show a spike in streaming frequencies close to zero. This is not the case for positive edges as followees need to release content more frequently for followers to want to follow them.
- Shared games, NMF cosine similarity, is-partner and common country are able to separate negative and positive edges while is-mature is not.
- NMF embeddings are able to create meaningful features that correspond to game genres or subtypes. This embedding overcomes the string-based encoding of games where different spellings can be used for the same videogame. Besides, this embedding is sparse which results in interpretable results. Sparsity also allows meaningful pairwise products between individual components to be used as standalone features.

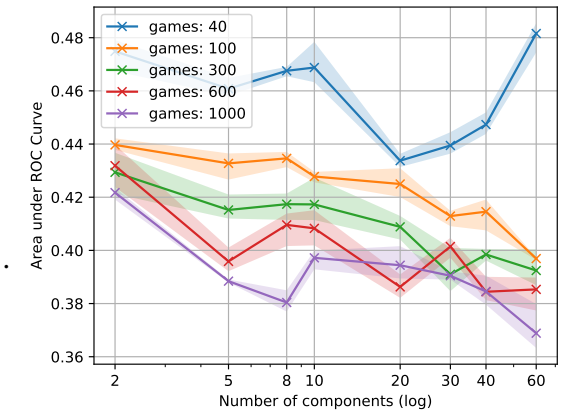

Figure 5.9: Euclidean similarity and standard encoding

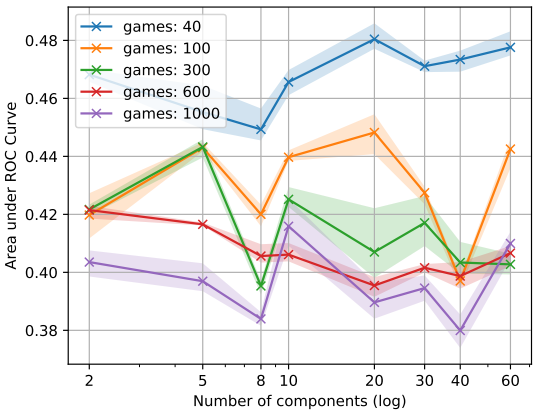

Figure 5.10: Cosine similarity and standard encoding

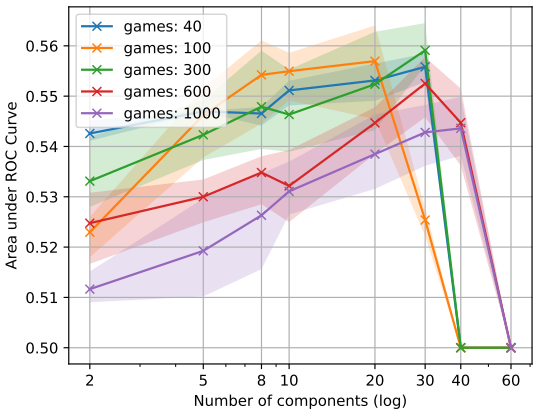

Figure 5.11: Euclidean similarity and TF-IDF encoding

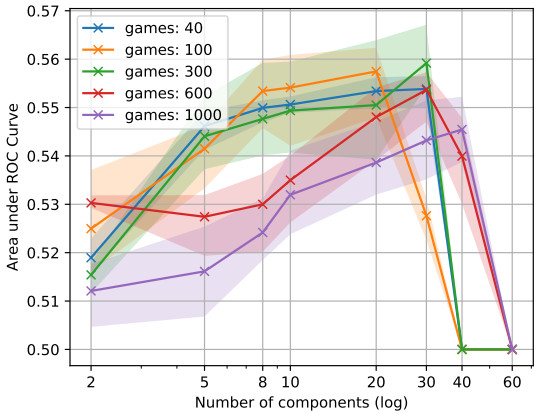

Figure 5.12: Cosine similarity and TF-IDF encoding

Link prediction using an LDA embedding for different distance metrics and data encodings. TF-IDF shows a dramatic improvement in performance over no TF-IDF. Smaller number of games results in better performance which shows that LDA might be a poor choice of embedding algorithm.

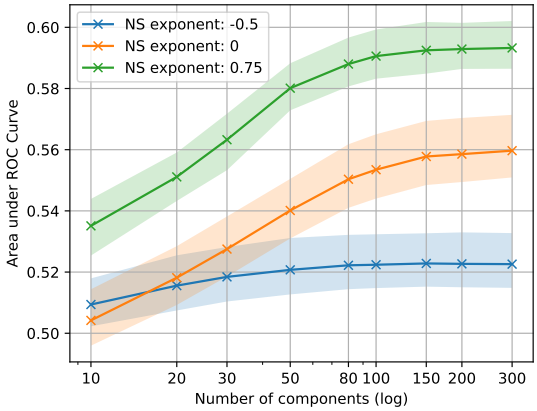

Figure 5.13: Link prediction using doc2vec embedding for different NS exponents and number of components. Cosine used as a similarity function. A NS exponent of 0.7 results in the best performance.

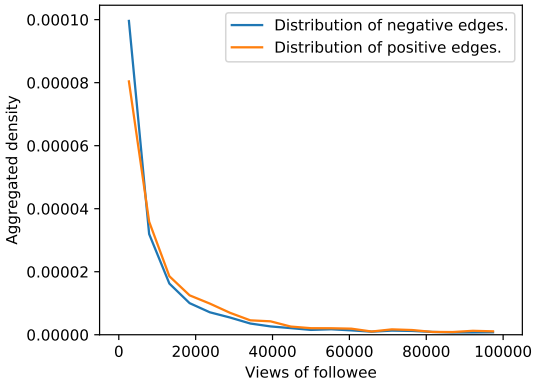

Figure 5.14: Distribution of positive and negative edges for aggregated views of the follower. No apparent separation between positive and negative edges.

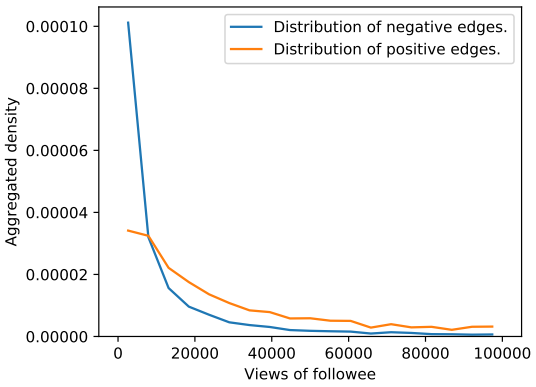

Figure 5.15: Distribution of positive and negative edges for aggregated views of the followee. The followee shows a uniformly higher number of aggregated views.

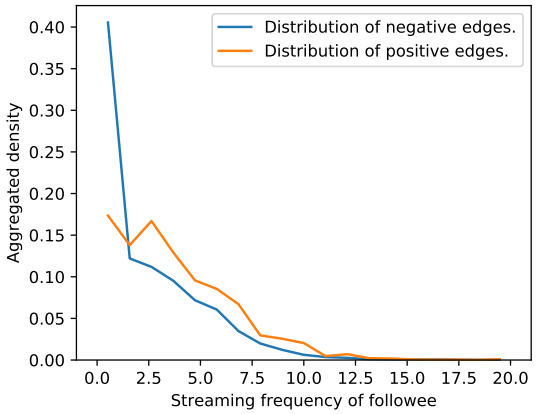

Figure 5.16: Distribution of positive and negative edges for followee's streaming frequence (number of streams overt the last 14 days). A larger number of followees in negative edges contained a frequency close to zero.

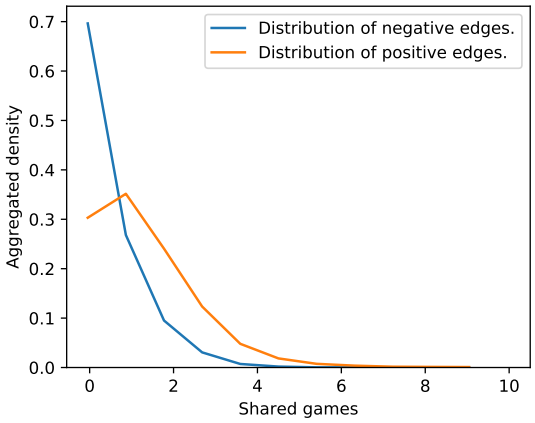

Figure 5.17: Distribution of positive and negative edges for shared number of games. Users in positive edges share a larger number of games.

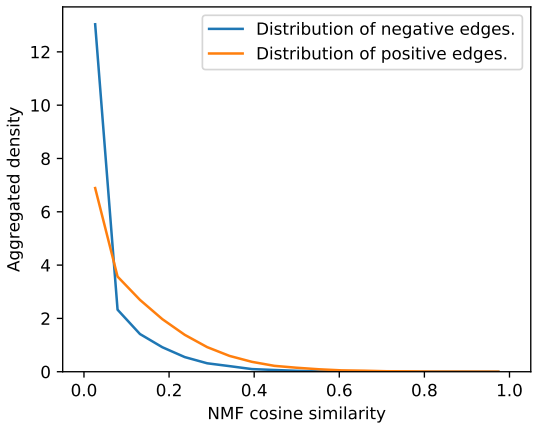

Figure 5.18: Distribution of positive and negative edges for different values of NMF cosine similarity. Positive edges have a generally higher cosine similarity than negative edges.

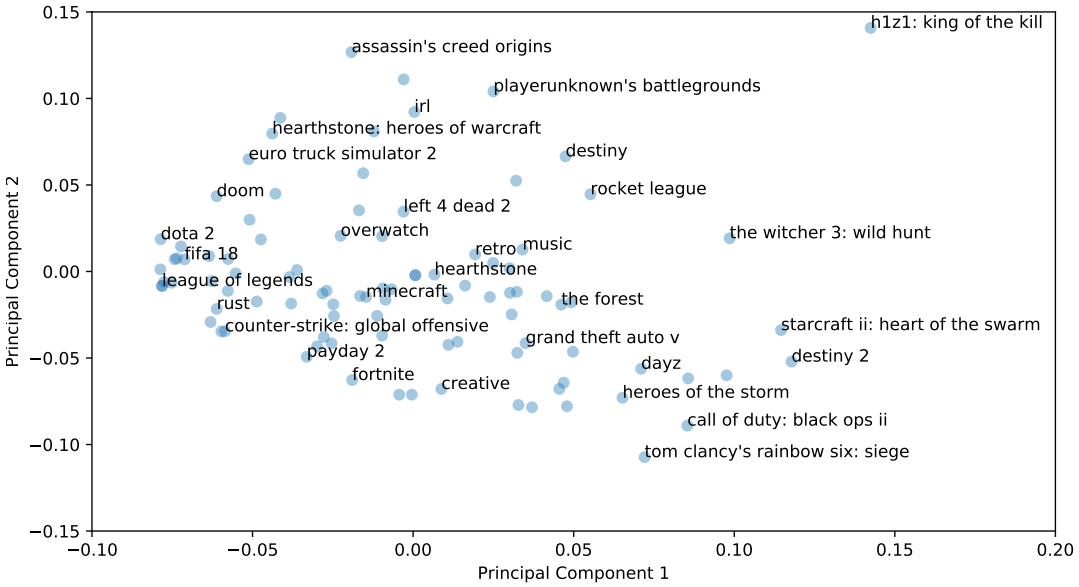

Figure 5.19: Top 100 games embedded in 2D space using NMF with the optimal paramteres as a factorization and PCA for visualization. Similar games are close to each other.

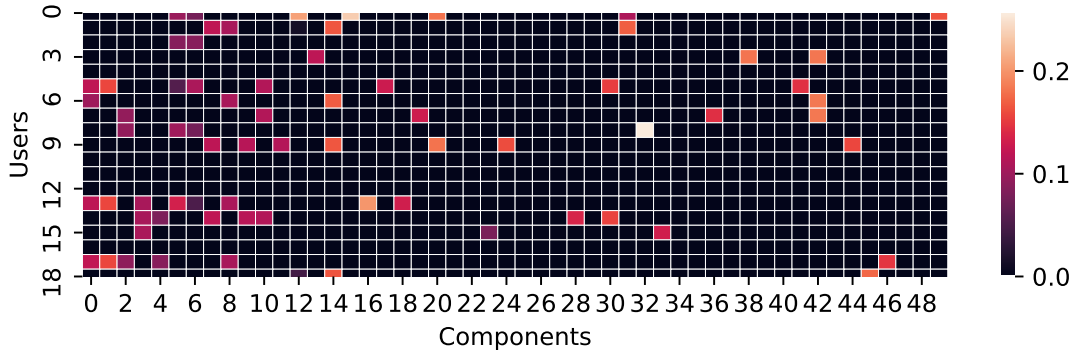

Figure 5.20: First 50 components for a sample of 20 users using a NMF-based feature embedding. Features are sparse.

Chapter 6

Combining features: the link predictor

In this section the topological and non-topological features are combined to assemble the link predictor algorithm. Three linear classifiers and a non-linear classifier were used. There are three reasons why I focused on linear classifiers despite the fact that non-linear classifiers allow more complexity in the classification:

- Throughout the inspection in section 4 and section 5, all features showed a linear distribution of negative and positive edges, i.e. for each feature positive edges were either generally larger in value or generally smaller in value.
- A fairly large number of features was used. This was mostly due to the 300 *pairwise NMF* components. Linear models are often recommended when the dimensionality of the data is very big, as the time complexity of non-linear models blows up for high dimensions.
- The most important reason is interpretability. Linear classifiers allocate coefficients (or weights) to each feature that represent the importance of the feature. By inspecting these weights one can understand how useful a feature is for the prediction task. All features were standardized to 0 mean and unit variance. Consequently, high negative values mean that small values of the feature result in positive instances, and high positive values mean that high values of the feature result in positive instances.

6.1 Feature evaluation

In this section the impact of each feature for link prediction is assessed by examining their corresponding model coefficients. Table 6.1 shows the SVM, LR and Perceptron coefficients for each feature. Four conclusions con be derived from these results:

- The impact of the features is quite polarized. Some features have very high coefficients while other have very small ones –close or even equal to zero. This means that some features like common neighbours, rooted PageRank or Aggregated NMF pairwise carry more meaning about the class of an edge than features like Jaccard Coefficient or channel views. However, another factor affecting this

	Weights		
Feature	SVM	LR	Perceptron
Common Neighbours (d)	**1.826** $\pm$ 0.632	**1.898** $\pm$ 0.291	**1.785** $\pm$ 0.148
Common Neighbours (u)	**2.167** $\pm$ 0.268	**3.516** $\pm$ 0.206	0.491 $\pm$ 0.045
Shortest Path (d)	-0.405 ± 0.094	**-0.565** $\pm$ 0.043	-0.032 ± 0.010
Shortest Path (d)	-0.206 ± 0.032	-0.114 ± 0.016	-0.011 ± 0.014
Rooted PageRank	**2.533** $\pm$ 1.168	**1.956** $\pm$ 0.803	**0.727** $\pm$ 0.078
Jaccard Coefficient (d)	0.096 ± 0.209	0.450 ± 0.082	-0.068 ± 0.020
Jaccard Coefficient (u)	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Adamic-Adar	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
In-Degree	-0.068 ± 0.123	0.074 ± 0.037	-0.024 ± 0.009
PageRank (d)	0.410 ± 0.014	0.249 ± 0.048	0.039 ± 0.005
PageRank (u)	0.341 ± 0.054	0.395 ± 0.129	0.026 ± 0.001
Clustering Coefficient	-0.068 ± 0.064	-0.115 ± 0.044	-0.019 ± 0.017
Betwenness Centrality	**0.883** $\pm$ 0.047	**1.190** $\pm$ 0.075	0.113 ± 0.014
Partner (follower)	0.012 ± 0.038	-0.013 ± 0.027	0.006 ± 0.001
Partner (followee)	0.121 ± 0.064	0.340 ± 0.024	0.033 ± 0.006
Mature (follower)	0.012 ± 0.038	-0.013 ± 0.027	0.006 ± 0.001
Mature (followee)	0.121 ± 0.064	0.340 ± 0.024	0.033 ± 0.006
Same Country	**1.184** $\pm$ 0.024	**1.858** $\pm$ 0.022	0.152 ± 0.031
Channel views (follower)	-0.039 ± 0.093	-0.096 ± 0.020	-0.041 ± 0.004
Channel views (followee)	0.077 ± 0.022	0.039 ± 0.032	0.018 ± 0.014
Shared games	**0.689** $\pm$ 0.124	**1.211** $\pm$ 0.059	0.211 ± 0.016
Streaming Frequency	0.130 ± 0.006	0.351 ± 0.018	0.036 ± 0.009
NMF overall cosine	0.085 ± 0.046	0.286 ± 0.011	-0.006 ± 0.021
Aggregated NMF pairwise	**15.525** $\pm$ 3.666	**17.949** $\pm$ 0.830	**10.931** $\pm$ 1.575

Table 6.1: Impact of individual features on link prediction for different classifiers (using the coefficient weights associated with each feature). High negative/positive values indicate higher importance. Weights with absolute value higher than 0.5 are in bold. d stands for directed and u for undirected.

result might be *synonym features*, i.e. features that refer to the same concept. PageRank, Betweenness Centrality and In-Degree are all measures of centrality, so the meaning that centrality carries to determine the class of an edge is split between these three. Besides, it can be the case that the feature that most accurately separates negative and positive edges of the three ends up receiving a higher weight while the others are ignored. This might explain why in-degree has absolutely no impact while Betweenness Centrality has a lot, as in-degree seemed to separate negative and positive edges when visually inspected in section 4.

- Weights are consistent across classifiers. This reinforces the notion of features being useful by themselves and not in the context of a classifier for a classifier's particular reasons. Consequently, conclusions about the usefulness of each feature can be extrapolated to a general setting.
- The Aggregated NMF pairwise feature is very high and NMF overall cosine is very small. The feature synonym effect might be a reason for this. Aggregated NFM pairwise encodes a much richer version of the information encoded by NMF overall cosine. Besides, Aggreggated NMF pairwise was calculated by summing the absolute value of the 300 pairwise features (individual weights shown in figure 6.1). The marginal noise in the weight of each feature can add up and might result in a disproportionately big coefficient. However, by inspecting figure 6.1, some pairwise features are indeed useful and indicate that the Aggregated NFM pairwise carries very useful meaning regarding the class of an edge.
- In short, the most useful features considering the weights of all three classifiers are Common Neighbours (directed and undirected), Shortest Path (directed), Rooted PageRank, PageRank, Betweenness Centrality, Same Country, Shared games, and Aggregated NMF pairwise. The most useless features are Jaccard Coefficient, In-Degree, Clustering Coefficient, Partner (follower), Mature (follower), Channel views, and NMF overall cosine.

6.1.1 NMF Pairwise and game genres

By looking at the individual NMF Pairwise features, insight can be derived into which NMF components are most useful at predicting a link between two users, and consequently what game genres are most useful. The individual weights of each NMF pairwise feature are represented in figure 6.1. Two conclusions can be drawn regarding the importance of the NMF features:

- The impact of the individual features is small compared to the most relevant topological and non-topological features. This makes sense as NMF encodings are sparse and most NMF-pairwise values are often 0 regardless of whether a link between users exists or not.
- Not all NMF pairwise features have the same importance. Most of them have relatively small values while a few of them have slightly higher values. Consequently, some NMF components are more useful at predicting the formation of

a link than others.

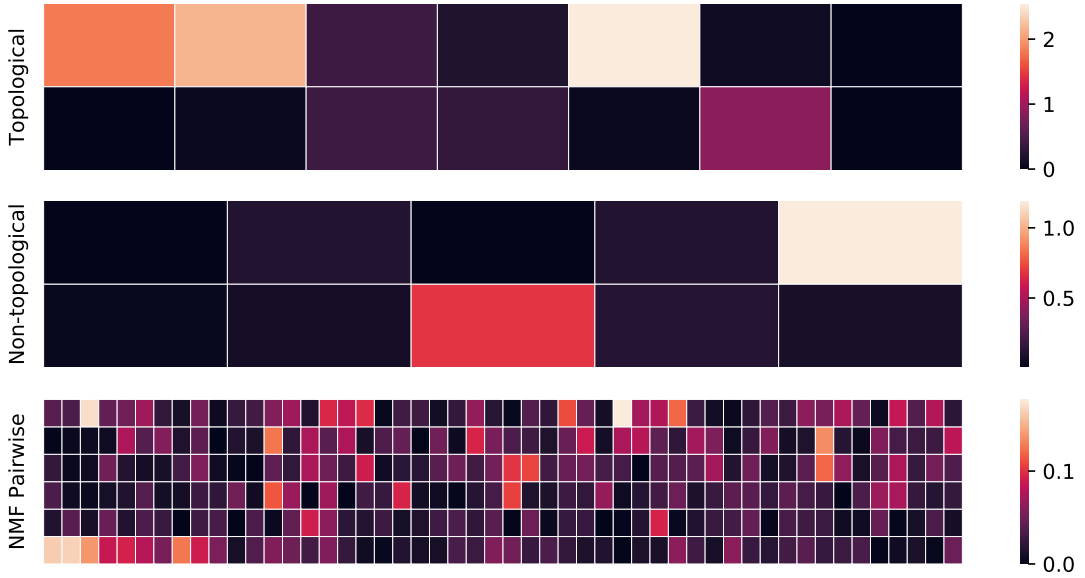

Figure 6.1: Absolute values of SVM-associated weights of different features. Higher weights indicate higher importance. NMF Pairwise features are variable on their importance. Topological and non-topological features included for comparability.

The top NMF pairwise features correspond to the NMF components number 281, 252 and 1. It was shown in section 5.2.2.4 how each component can be interpreted as a game feature or genre. These components respectively correspond to: *arcade/ old-school games*, *battle action games*, and *online multiplayer battle games* (the games representing each genre are shown in table 5.3).

Table 6.2 shows the top NMF components according to their SVM weight and the genre associated with them. Being able to interpret what each component means is very insightful information. When the genres associated with the top components are shared between two users the formation of a link between them is more likely. This information could be very useful for Twitch in order to improve content suggestion. Users and games that involve genres with high weights are more likely to draw attention to users interested in those genres.

It's interesting to note that the meaning of the components in table 6.2 doesn't fully overlap, i.e. each component represents a slightly different genre. However, it can be seen that battle-related games are very useful indicators of link formation. Battle-related games are one of the top genres together with MOBAs streamed in Twitch [16]. Thus, such a common genre is likely to divide the users significantly on whether they like it or not, rendering it useful for link prediction. Other more niche genres like arcade or simulation games also have high weights. This might be because of the tighter, less mainstream communities that arise from these games. Users in tight non-topological communities are more likely to form links between them.

Component no.	Game genre	SVM Weight
281	Arcade / old-school games	0.177 ±0.080
252	Battle action games	0.171 ±0.171
1	Online multiplayer battle games	0.165 ±0.055
0	Simulation games	0.163 ±0.041
2	Battle royal games	0.139 ±0.021
242	Grand Theft Auto Franchise	0.135 ±0.086
7	Mega Man franchise	0.127 ±0.067
212	Games for younger audiences	0.126 ±0.080
284	Action assassin games	0.120 ±0.057
192	Fallout franchise	0.120 ±0.156

Table 6.2: Genres of the top 10 NMF components according to their SVM weight. Each component represents a differnt genre. Higher SVM weight mean higher importance in the link prediction decision.

6.2 Evaluation of binary classifiers

In this section, the different binary classifiers are evaluated using their Area under the ROC Curve. The linear classifiers used are Linear Support Vector Machines, Perceptron and Logistic Regression. A small L2 regularization of 0.0001 was used for all models. One non-linear classifier was used: RBF Support Vector Machines. The optimal performance of RBF SVM heavily relies on the joint exploration of the regularization strength (C) and the impact of individual datapoints in the RBF kernel function (γ). Large values of C result in all training datapoints being properly classified and large values of gamma defining the influence of a single datapoint [44].

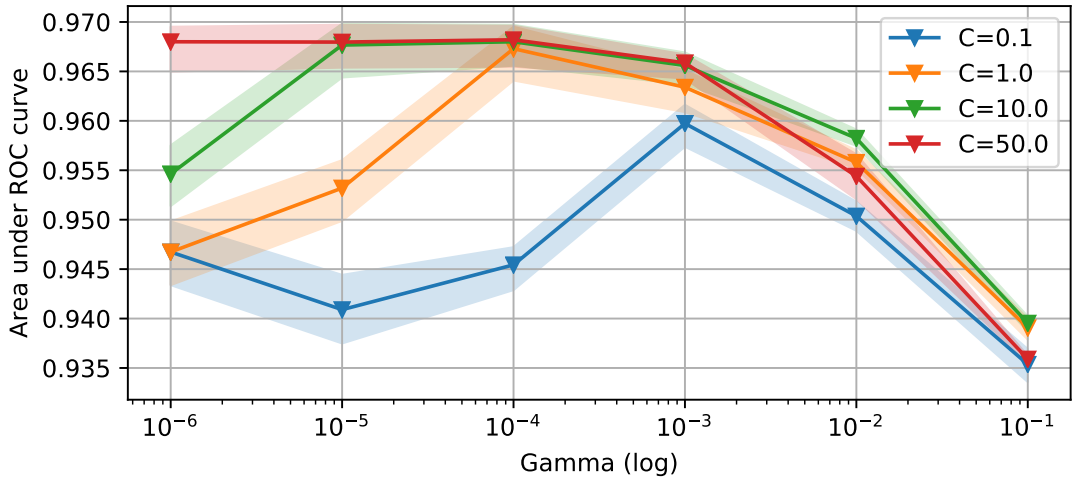

Figure 6.2: Performance of link prediction for RBF SVM with different degrees of regularization (C) and variations of the kernel function (γ). $C = 50.0, \gamma = 10^{-4}$ results in the best performance.

The results of the C and γ parameter search can be seen in figure 6.2. Smaller degrees of regularization (large values of C) result in better performances. The best combination

	Linear SVM	RBF SVM	Perceptron	LR
All	0.964 ±0.002	0.968 ±0.003	0.937 ±0.002	0.968 ±0.001
Topological	0.943 ±0.002	0.935 ±0.004	0.885 ±0.044	0.931 ±0.004
Non-topological	0.940 ±0.005	0.950 ±0.003	0.898 ±0.005	0.949 ±0.002

Table 6.3: Performance of different classifiers for link prediction using different sets of features. Values indicate the area under the ROC curve for each classifier. RBF SVM and LR are the best models when all features are used, linear SVM the best for topological features, and RBF SVM is the best model for non-topological features.

of C and γ is $C = 50.0, \gamma = 10^{-4}$ where the Area under the ROC curve is 0.968 ± 0.003. Consequently, these settings were used for the RBF SVM in the remainder of the experiments.

Table 6.3 shows the classification results for the different models. It's worth keeping in mind that a random classification corresponds to an area under the ROC curve of 0.5, so values over that figure are better than a random baseline. RBF SVM and LR are the best best models when all features are used, linear SVM is the best model for topological features, and RBF SVM is the best model for non-topological features. It's interesting to notice that a linear classifier (Linear SVM) outperforms a non-linear classifier (RBF SVM) when only topological features are used. This hints that there are not many interrelationships between features but rather each feature has a true linear impact. This is not the case for non-topological features, probably arising from the complex nature of the NMF-pairwise features that do have meaningful interrelations between them. Overall, linear and non-linear classifiers are able to exploit the features to a very similar extent.

Non-topological features seemed to be more useful than topological ones, as RBF SVM, Perceptron, and LR obtained better performance using non-topological features exclusively over topological ones. Nevertheless, the combination of topological and non-topological features resulted in the best performance for all classifiers.

In order to frame this result within the context of other feature-based link prediction algorithms, it's worth comparing the obtained best area under ROC score with the winner of The 2011 IJCNN Social Network challenge [15], who achieved a score of 0.970 (compared to the 0.968 obtained here). This score was achieved on a completely different dataset (Flickr). However, it serves as a ballpark figure to understand how good the obtained results are.

Chapter 7

Conclusion

Throughout this project, I started by introducing feature-based supervised link prediction, a comprehensive set of previously used topological and non-topological features, different binary classifiers, and embedding methods to exploit non-topological information. Relevant work on the problem of link prediction was reviewed, with an emphasis on feature-based supervised link prediction.

The work carried out contributes to the link prediction literature by exploring supervised feature-based link prediction on a not-yet-explored class of graphs where content type is shared between users. Experiments were carried out using a subset of the Twitch social network. Non-Negative Matrix factorization paired with cosine similarity was found to be the most effective embedding to exploit the non-topological shared content, resulting in an area under the ROC curve of 0.701 ± 0.004 for link prediction. The obtained NMF embeddings represent meaningful sparse features that correspond to interpretable genres. This is particularly useful as it throws insight into the gaming behaviour of users and fixes the user-input string-based encoding of video games. A small subset of the pairwise NMF features have a salient impact in link prediction. These features represent both very popular genres and more niche ones. The genre that carries the most meaning about the link between two users is *arcade / old-school games*.

The impact of the different topological and non-topological features is polarized. The most useful features according to the classifier coefficient weights are (from most to least useful) Aggregated NMF pairwise, rooted PageRank, common neighbours, same country, Betweenness Centrality and shared games. All these features resulted in coefficient weights of at least 0.7 for all the linear classifiers used. Consequently, neighbourhood overlap, distance-based metrics and centrality features were an effective metric to capture the presence of a link between users. Measures of local clustering however were not an effective metric.

Four binary classifiers were explored, namely Linear SVMs, RBF SVMs, Perceptron and Logistic Regression. Logistic Regression and RBF SVM resulted in the best performance with an area under the ROC curve of 0.968 ± 0.003. Linear SVMs resulted in the best performance when only topological features were used. Exclusively us-

ing non-topological features resulted in a better performance than exclusively using topological features. Overall, linear and non-linear classifiers resulted in similar performances for the task of link prediction.

Bibliography

[1] 25 useful twitch statistics for influencer marketing managers. https://influencermarketinghub.com/twitch-statistics/. Accessed: 2020-04-07.

[2] Bibliographic database, elsevier biobase - current awareness in biological sciences (cabs).

[3] Dblp computer science bibliography. https://dblp.uni-trier.de/.

[4] Stream elements. https://streamelements.com/.

[5] Twitch. https://www.twitch.tv/.

[6] Twitch statistics & charts. https://twitchtracker.com/statistics. Accessed: 2020-04-15.

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

[8] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*, volume 30, pages 798–805, 2006.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[10] Stephen P Borgatti. Centrality and network flow. *Social networks*, 27(1):55–71, 2005.

[11] Kendrick Boyd, Kevin H Eng, and C David Page. Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 451–466. Springer, 2013.

[12] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. 1998.

[13] Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. Word2vec applied to recommendation: Hyperparameters matter. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 352–356, 2018.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] William Cukierski, Benjamin Hamner, and Bo Yang. Graph-based features for supervised link prediction. In *The 2011 International Joint Conference on Neural Networks*, pages 1237–1244. IEEE, 2011.

[16] Jie Deng, Felix Cuadrado, Gareth Tyson, and Steve Uhlig. Behind the game: Exploring the twitch streaming platform. In *2015 International Workshop on Network and Systems Support for Games (NetGames)*, pages 1–6. IEEE, 2015.

[17] Yuxiao Dong, Jie Tang, Sen Wu, Jilei Tian, Nitesh V Chawla, Jinghai Rao, and Huanhuan Cao. Link prediction and recommendation across heterogeneous social networks. In *2012 IEEE 12th International conference on data mining*, pages 181–190. IEEE, 2012.

[18] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[19] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural computation*, 23(9):2421–2456, 2011.

[20] Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach, and Yuval Elovici. Link prediction in social networks using computationally efficient topological features. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 73–80. IEEE, 2011.

[21] Jerome H Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, 1997.

[22] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[23] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear Algebra*, pages 134–151. Springer, 1971.

[24] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. 2009.

[25] William A Hamilton, Oliver Garretson, and Andruid Kerne. Streaming on twitch: fostering participatory communities of play within live mixed media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1315–1324, 2014.

[26] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[27] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[28] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[29] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 61–68, 2009.

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

[31] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6. Siam, 1998.

[32] Rong-Hua Li, Jeffrey Xu Yu, and Jianquan Liu. Link prediction: the power of maximal entropy random walk. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1147–1156, 2011.

[33] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[34] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252, 2010.

[35] Haifeng Liu, Zheng Hu, Hamed Haddadi, and Hui Tian. Hidden link prediction based on node centrality and weak ties. *EPL (Europhysics Letters)*, 101(1):18004, 2013.

[36] Zhengdong Lu, Berkant Savas, Wei Tang, and Inderjit S Dhillon. Supervised link prediction using multiple sources. In *2010 IEEE international conference on data mining*, pages 923–928. IEEE, 2010.

[37] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[38] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

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

[40] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

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

[42] Mark EJ Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[43] Mark EJ Newman and Juyong Park. Why social networks are different from other types of networks. *Physical review E*, 68(3):036122, 2003.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[45] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[46] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[47] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *arXiv preprint arXiv:1909.13021*, 2019.

[48] G Salton and MJ McGill. Introduction to modern information retrieval mcgraw hill book company. *New York*, 1983.

[49] Ramesh R Sarukkai. Link prediction and path analysis using markov chains. *Computer Networks*, 33(1-6):377–386, 2000.

[50] Sucheta Soundarajan and John Hopcroft. Using community information to improve the precision of link prediction methods. In *Proceedings of the 21st International Conference on World Wide Web*, pages 607–608, 2012.

[51] Jorge Valverde-Rebaza and Alneu de Andrade Lopes. Exploiting behaviors of communities of twitter users for link prediction. *Social Network Analysis and Mining*, 3(4):1063–1074, 2013.

[52] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 322–331. IEEE, 2007.

[53] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[54] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[55] Yun Xue, Chong Sze Tong, and Tiechen Li. Evaluation of distance measures for nmf-based face image applications. *JCP*, 9(7):1704–1711, 2014.

[56] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015.

[57] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International conference on algorithmic applications in management*, pages 337–348. Springer, 2008.