Recommending Teammates with Deep Neural Networks*

Palash Goyal
USC Information Sciences Institute
Marina del Rey, CA, USA
goyal@isi.edu

Anna Sapienza
USC Information Sciences Institute
Marina del Rey, CA, USA
annas@isi.edu

Emilio Ferrara
USC Information Sciences Institute
Marina del Rey, CA, USA
ferrarae@isi.edu

ABSTRACT

The effects of team collaboration on performance have been explored in a variety of settings. Online games enable people with significantly different skills to cooperate and compete within a shared context. Players can affect teammates' performance either via direct communication or by influencing teammates' actions. Understanding such effects can help us provide insights into human behavior as well as make team recommendations. In this work, we aim at recommending teammates to each individual player for maximal skill growth. We study the effect of collaboration in online games using a large dataset from *Dota 2*, a popular Multiplayer Online Battle Arena game. To this aim, we construct an online co-play teammate network of players, whose links are weighted based on the gain in skill achieved due to team collaboration. We then use the performance network to devise a recommendation system based on a modified deep neural network autoencoder method.

CCS CONCEPTS

• **Information systems → Massively multiplayer online games**; **Data mining**; • **Computing methodologies → Neural networks**; **Factorization methods**;

KEYWORDS

recommendation system, link prediction, deep neural network, graph factorization, multiplayer online games, team formation

ACM Reference Format:
Palash Goyal, Anna Sapienza, and Emilio Ferrara. 2018. Recommending Teammates with Deep Neural Networks. In *HT '18: 29th ACM Conference on Hypertext and Social Media, July 9–12, 2018, Baltimore, MD, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3209542.3209569

1 INTRODUCTION

How is the performance of an individual affected by the interaction with peers? Psychologists and sociologists have researched this question for several decades in various contexts. Effect of coworkers on productivity at work [18], influence of students in a classroom [22], teammates effect in sports [5] are a few examples of such studies. These studies provide methods to measure influence in a collaborative setting and provide insights into the positive and negative effect of collaborations in various fields.

The advent of online games has opened new directions to explore. They provide unique challenges as the players often have minimal physical communication. The players collaborate in the virtual environment, the dynamics of which are often vastly different from the settings studied previously [14]. In online games, people from all over the world with varying backgrounds cooperate to achieve a common goal. Thus, they provide a suitable environment to analyze collaboration with higher diversity in individuals.

In this paper, we study a class of online games, called Multiplayer Online Battle Arena (MOBA) games, such as League of Legends (LoL), Defense of the Ancients 2 (Dota 2), Heroes of the Storm, and Paragon. In these games, two teams compete against each other to destroy the other team's base. At the start of the match each player chooses a character with a certain role and abilities. Broadly, the roles can be defensive or attacking. Players with defensive roles protect and cooperate with attacking players in a fight with the aim of moving forward towards the opponent's base. Therefore, both cooperation and influence of other players are intrinsic mechanisms of this genre of games, and can be either explicit or implicit: players can communicate through chat to shape the teammates' actions or can choose their moves based on their teammates' actions.

Prior research has shown that cooperation plays a key role in obtaining a high win rate in MOBA games [8, 24]. However, multiple factors can affect both cooperation and performance in this type of games. A great amount of work has been devoted to identify such factors, which can be broadly classified into two categories: 1) studies which use social interactions such as effect of friendship [17, 19] and frequency of co-play [15], and studies which focus on each player's choices, such as strategy, role, session length, and location of play [8, 9, 20, 24]. These studies show how each of these factors can have a significant impact on performance.

Although significant research has been done on identifying factors influencing a team's performance in MOBA games, the impact of such factors, and in particular the influence of cooperation on an individual's skills, needs further exploration. In the present work, we take a step in this direction and aim to understand and predict this impact. First, we note that performance of an individual and a team can greatly vary as players may have varying contribution to the outcome of a match. Thus, using the team's performance as measure for an individual can be misleading. Here, we thus take into account the individual skill level which is computed on the player's attributes and in-game performance. By using this measure of a player's performance, we study the effect of other players on the skill level of an individual over multiple matches in the well-known MOBA game Dota 2. To this aim, we construct a directed co-play network of players, the links of which exist if two players were

*PG and AS contributed equally to this work.

HT '18, July 9–12, 2018, Baltimore, MD, USA

ACM ISBN 978-1-4503-5427-1/18/07…$15.00
https://doi.org/10.1145/3209542.3209569

teammates in the matches. Each directed link (u, v) is weighted based on the increase/decline of skill level of player v when playing with u. The network thus encompasses the cooperation effect aggregated over multiple matches. We finally study such a network and build a recommendation system for players. To this aim, we devise a modified deep neural network autoencoder that is able to predict the influence of teammates on each player.

In our experiments, we show that our teammate recommendation system can learn the inherent structure of the network and predict with high accuracy the teammates that are more beneficial to each player. To show this, we evaluate our autoencoder against a baseline model on the tasks of predicting an individual's skill gain and making player recommendations. We show that our model significantly outperforms the baseline and achieves an improvement of 9.00% on the former task. For player recommendations, the gain is instead 19.50%. We also compare our model against the widely used graph factorization model which does not achieve any significant gain over baseline. This shows that non-linear models are fundamental to capture the structure of co-play networks.

2 DATA COLLECTION AND PREPROCESSING

In the present work, we analyze data of Defense of the Ancient 2 (Dota 2), a popular MOBA game released in July 2013. We used *OpenDota* [7] to scrape the matches of Dota 2 in 2015. The overall dataset consists of 3,300,146 matches and 1,805,225 players. The game allows players to practice and play in different game modes. For the purpose of this study, we need to consider all the matches involving just human players. Therefore, we select the so called "Ranked" and "Public" matches. We further process this data to select matches for which there is no missing information. This step is done to avoid noisy and incomplete data samples. Finally, to understand the influence of teammates on an individual's performance, we compute the skill of players after each match in their history. Dota 2 has an internal matchmaking rank (MMR), which is a score defining each player skill that is updated any time the player conclude a match. However, the MMR is not disclosed by Dota 2 and thus, we use a proxy for players' skill level. Here, we use the TrueSkill [11] matchmaking system, developed by Microsoft. The TrueSkill value, as the Dota 2 MMR, has been designed for multiplayer team-based games such as MOBA games. It models players' skill through two values: the skill level and its certainty (i.e. a confidence interval for the skill level). The skill level changes accordingly to players performance in each match, while its confidence interval shrinks with the number of matches in the player's history. For a game setting like the one in Dota 2 (two teams of five players), the Trueskill requires at least 46 matches to accurately model the skill for a player.[1] Therefore, we further discard those players in our dataset having less than 46 matches. The preprocessing procedure leads to a final dataset consisting of all matches played by 87,155 players.

3 NETWORK GENERATION

We now explain the approach used to construct the co-play performance network. Here, nodes of the co-play network represent players, while links reflect the TrueSkill score variations depending on the each player's teammate and aggregated over time.

Let us consider the set of 87, 155 players in our post-processed Dota 2 dataset, and the related matches they played. For each player p, we define the player history to be the temporally ordered set $M_p = [m_0, m_1, \cdots, m_N]$ of matches played by p. As we consider just Public and Ranked matches in our dataset, matches are always composed by two opposing teams of 5 human players each, thus each match $m_i \in M_p$ can be further identified by a 4-tuple (t_1, t_2, t_3, t_4) of player's teammates.

Given a teammate t of player p in match $m_i \in M_p$, we define his/her influence on the performance of player p as the weight:

$$w_{pt,m_i} = ts_{m_i} - ts_{m_{i-1}}, \tag{1}$$

where, ts_{m_i} is the TrueSkill score of the player p after match $m_i \in M_p$. Thus, the weight w_{pt,m_i} captures the performance gain/loss of player p when playing with a given teammate t. This step generates as a result a set of directed links connecting together the players to their teammates in each match, and being characterized by the relative weights based on the fluctuations of TrueSkill level of players.

Next, we build the overall Performance Network, by aggregating the links connecting each couple of teammates in the network. Thus, the resulting Performance Network has a link between two nodes if the corresponding players were teammates at least once in the total temporal span of our dataset. Each link has a weight that is now defined as the sum of the weights given in (1). Thus, given player p and any possible teammate t in the network, their aggregated weight w_{pt} is equal to

$$w_{pt} = \sum_{i=0}^{N} w_{pt,m_i}, \tag{2}$$

where $w_{pt,m_i} = ts_{m_i} - ts_{m_{i-1}}$ if $t \in m_i$, and 0 otherwise. The resulting network has 87, 155 nodes and 4, 906, 131 directed links with weights $w_{pt} \in [-0.58, 1.06]$.

The co-play performance network thus built may have unreliable weights. If two players play together just few times, the confidence we have on the corresponding weight is low. For example, if two players are teammates just one time their final weight only depends on that unique instance, and thus might lead to biased results. To tackle this issue, we further compute the distribution of the number of occurrences a couple of teammates plays together in our network and set a threshold based on these values. In particular, we decided to retain only pairs that played more than 2 matches together.

Finally, as many node embedding methods require a connected network as input [2], we extract the Largest Connected Component (LCC) of the performance network, which will be used for the performance prediction and evaluation. In particular, the LCC includes 38, 563 nodes and 1, 444, 290 links. Consistently with the overall network, the weights of the LCC range from -0.58 to 1.06.

4 PERFORMANCE PREDICTION

In the following, we test whether the co-play performance network presents an underlying structure that can be leveraged to predict the performance gain of players with unknown teammates. To this aim, we evaluate both linear and non-linear models against a

[1] https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/

baseline and show that our model achieves better performance in recommending teammates that maximize the skill of a player.

4.1 Problem Formulation

Let us consider the co-play performance network $G = (V, E)$ with weighted adjacency matrix W. A weighted link (i, j, w_{ij}) denotes that player i gets a performance variation of w_{ij} after playing with player j. We can formulate the recommendation problem as follows. Given an observed instance of a co-play performance network $G = (V, E)$ we want to predict the weight of each unobserved link $(i, j) \notin E$ and use this result to further recommend teammates to each player $i \in V$ by predicting the ranking of all other players $j \in V$ $(\neq i)$.

4.2 Network Modeling

To predict the unobserved weights and recommend teammates to each player in our network, we assume that the co-play network we created contains information and hidden structures that can be used to forecast the influence, and thus the skill gain, of unseen pairs of players. To verify this assumption, we need a model that is able to capture such underlying structures.

To this aim, we modify a deep neural network autoencoder and we test its predictive power against two classes of approaches (both linear and non-linear) that are widely applied to build recommendation systems: (a) factorization based [1, 13, 21], and (b) deep neural network based [6, 12, 23].

4.2.1 Factorization. Graph Factorization models for directed networks aim at finding a low-dimensional representation of the weighted adjacency matrix $\mathbf{W}$. The adjacency matrix is indeed decomposed in two matrices $\mathbf{U} \in \mathbb{R}^{n \times d}$ and $\mathbf{V} \in \mathbb{R}^{n \times d}$ with number of hidden dimensions d such that the following function is minimized

$$f(\mathbf{U}, \mathbf{V}) = \sum_{(i,j) \in E} (w_{ij} - < \boldsymbol{u_i}, \boldsymbol{v_j} >)^2 + \frac{\lambda}{2}(\|\boldsymbol{u_i}\|^2 + \|\boldsymbol{v_j}\|^2)$$

Note that the sum is computed over the observed links to avoid of penalizing the unobserved one as overfitting to 0s would affect the predictions. Here, λ is chosen as a regularization parameter to give preference to simpler models for better generalization.

4.2.2 Traditional Autoencoder. Autoencoders are unsupervised neural networks used to minimize the loss between reconstructed and input vectors. A traditional autoencoder is made of two main parts: (a) an encoder, which maps the input vector into low-dimensional latent variables; and, (b) a decoder, which maps the latent variables to an output vector. The reconstruction loss can be written as

$$L = \sum_{i=1}^{n} \|(\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x_i})\|_2^2, \tag{3}$$

where $\boldsymbol{x_i}$s are the inputs and $\hat{\boldsymbol{x}}_{\boldsymbol{i}} = f(g(\boldsymbol{x_i}))$. $f(.)$ and $g(.)$ are the decoder and encoder functions respectively. Deep autoencoders have recently been adapted to the network setting [6, 12, 23]. An algorithm proposed by *Wang et al.* [23] jointly optimizes the autoencoder reconstruction error and Laplacian Eigenmaps [4] error to learn representation for undirected networks. However, the "Traditional Autoencoder" equally penalizes observed and unobserved links in the network, while the model adapted to the network setting cannot be generalized to the case of directed networks. Thus, we propose to modify the Traditional Autoencoder model to both take into account directed links and be able to find a low-dimensional representation in a network setting.

4.2.3 Teammate Autoencoder. To model directed networks, we propose a modification of the Traditional Autoencoder model, that takes into account the adjacency matrix representing the directed network. Moreover, in this formulation we only penalize the observed links in the network, as our aim is to predict the weight and the corresponding ranking of the unobserved links. We then write our "Teammate Autoencoder" reconstruction loss as:

$$L = \sum_{i=1}^{n} \|(\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x_i}) \odot [a_{i,j}]_{j=1}^{n}\|_2^2, \tag{4}$$

where $a_{ij} = 1$ if $(i, j) \in E$, and 0 otherwise. Here, x_i represents i^{th} row the adjacency matrix and n is the number of nodes in the network. Minimizing this loss functions yields the neural network weights $\mathbf{W}$ and the learned representation of the network $\mathbf{Y} \in \mathbb{R}^{n \times d}$.

4.3 Evaluation Framework

4.3.1 Experimental Setting. To evaluate the performance of the models on the task of teammates' recommendation, we use a cross-validation framework. We randomly "hide" 20% of the weighted links and use the rest of the network to learn the embedding, i.e. the low-dimensional representation, of each player in the network. We then use each player's embedding to predict the weights of the unobserved links. As the number of player pairs is too large, we evaluate the models on multiple samples of the co-player performance networks (similar to [10, 16]) and report the mean and standard deviation of the used metrics. Instead of uniformly sampling the players as performed in [10, 16], we use random walks [3] with random restarts to generate sampled networks having similar degree and weight distributions as the original network.

4.3.2 Evaluation Metrics. We use Mean Squared Error (*MSE*) and Mean Absolute Normalized Error (*MANE*) as evaluation metrics. *MSE* evaluates the accuracy of the predicted weights, whereas *MANE* evaluates the ranking obtained by the model.

First, we compute *MSE* to evaluate the error in the prediction of weights. It is defined as

$$MSE = \|\boldsymbol{w}^{test} - \boldsymbol{w}^{pred}\|^2 ,$$

where $\boldsymbol{w}_{test}$ is the list of weights of links in the test subnetwork, and $\boldsymbol{w}_{pred}$ is the list of weights predicted by the model.

Second, to test the models' recommendations for each player, we define the Mean Absolute Normalized Error (*MANE*), which computes the normalized difference between predicted and actual ranking of the test links among the observed links and averages over the nodes. Formally, it can be written as

$$MANE = \frac{\sum_{i=1}^{|V|} MANE(i)}{|V|} ,$$

where $MANE(i) = \frac{\sum_{j=1}^{|E_i^{test}|} \left| rank_i^{pred}(j) - rank_i^{test}(j) \right|}{|E_i^{train}||E_i^{test}|}$ and $rank_i^{pred}(j)$ represents the rank of the j^{th} vertex in the list of weights predicted for the player i.

4.4 Results and Analysis

Here, we report the results achieved by the Graph Factorization, the Traditional Autoencoder and our Teammate Autoencoder. To this aim we first analyze the models' performance on the co-play performance network with respect to the MSE measure. In Fig. 1, we compare the models against an "average" baseline, where we compute the average performance of the players' couples observed in the training set and use it as a prediction for each hidden teammate link.

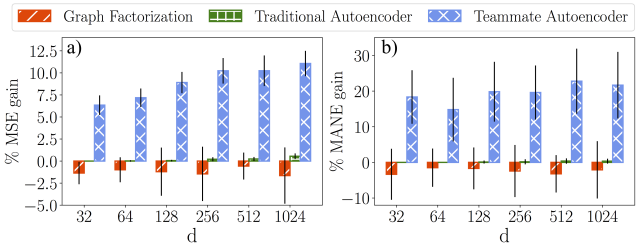

Figure 1: Performance results. a) Mean Squared Error (*MSE*) gain of models over average prediction. b) Mean Absolute Normalized Error (*MANE*) gain of models over average prediction.

Table 1: Mean and standard deviation of player performance prediction (*MSE*) and teammate recommendation (*MANE*) for $d = 1,024$ in the performance network.

	Baseline	GF	Autoencoder	Teammate Autoencoder
MSE	4.55/0.14	4.59/0.17	4.54/0.15	**4.15/0.14**
MANE	0.078/0.02	0.081/0.02	0.074/0.01	**0.059/0.008**

Fig. 1a shows the variation of the percentage of the *MSE* gain (average and standard deviation) while increasing the number of latent dimensions d for in each model. We can observe that the Graph Factorization model generally performs worse than the baseline, with values in the range $[-1.64\%, -0.56\%]$ and average of -1.2%. This suggests that the performance network of Dota 2 requires non-linearity to capture its underlying structure. Nevertheless, a traditional non-linear model is not enough to outperform the baseline. The Traditional Autoencoder reaches indeed marginal improvements: its gain over the baseline is in the range of $[0.0\%, 0.55\%]$ and it reaches an average gain of 0.18%. On the contrast, our Teammate Autoencoder achieves substantial gain over the baseline across the whole spectrum. Moreover, its performance generally increases for higher dimensions, thus retaining more structural information. The MSE gain for different dimensions over the baseline of the Teammate Autoencoder spans between 6.34% and 11.06%, with an average gain over all dimensions of 9.00%. We also computed the MSE average over 10 runs and $d = 1,024$, shown in Tab. 1. We can observe that the value decreases from the baseline prediction of 4.55 to our Teammate Autoencoder prediction of 4.15.

We finally compare the models' performance in providing individual recommendations by analyzing the *MANE* metric. Fig. 1b shows the percentage of the *MANE* gain for different dimensions computed against the average baseline for the performance network. Analogously to the *MSE* case, the Graph Factorization performs worse than the baseline. Its values range indeed between -3.34% and -1.48%, with an average gain of -2.37%, despite the increment in the number of dimensions. The Traditional Autoencoder achieves marginal gain over the baseline for dimensions higher than 128 ($[0.0\%, 0.37\%]$), and has an average gain over all dimensions of 0.16%. Our model attains instead significant percentage gain in individual recommendations over the baseline. It achieves an average percentage of *MANE* gain spanning from 14.81% to 22.78%, with an overall average of 19.50%. It is worth noting that the performance in this case does not monotonically increase with dimensions. This might imply that for individual recommendations the model overfits at higher dimensions. We report the average value of *MANE* in Tab. 1 for $d = 1,024$. Our model obtains average values of 0.059, compared to 0.078 of the average baseline.

In conclusion, our model achieves high performance on both tasks of predicting the unobserved weights of a co-play performance network and of recommending teammates that will have high impact in the skill gain of players, thus maximizing their performance when cooperating in a match.

5 CONCLUSIONS

In this work, we studied the influence of teammates on a player's performance in online games. Specifically, we achieved three goals: (i) we quantified the individual performance gain in online collaborative setting and studied the co-play network; (ii) we provided a framework to recommend teammates which maximize individual skill gain; and (iii) we illustrated the utility of a deep learning based recommendation model over linear models for this task.

For this purpose, we performed experiments on a popular Multiplayer Online Battle Arena game called Dota 2. We preprocessed the data to extract useful and robust information from the matches played in 2015. We used the Trueskill measure to track each individual's performance over matches and constructed a co-play performance network with players as nodes and aggregated gain in skill due to collaboration as the links' weights. Based on the network weights, we observed that players can have both adverse and positive effect on teammate's performance.

Thus, we developed a recommendation system for the co-play network based on a deep learning model. The model predicts performance gains for a set of players and uses the predictions to accurately recommend players which maximize skill gain of an individual. We compared our method against an average recommendation baseline, a traditional neural network autoencoder and a state-of-the-art graph factorization model. We showed our modified deep autoencoder model significantly outperforms the other models. Furthermore, we observed that the factorization based model only marginally improves over the average baseline showcasing the necessity of a complex non-linear model.

In the future, we plan to study the long term effect of a teammate on an individual's performance and evaluate our model on other online games and platforms. We also intend to test the model in real settings to directly evaluate the effect of our recommendations.

Acknowledgements. The authors are grateful to DARPA for support (grant #D16AP00115). This project may not reflect the position/policy of the Government; no official endorsement should be inferred. Approved for public release; unlimited distribution.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *Proc. 22nd international conference on World Wide Web*. ACM, 37–48.

[2] Nesreen K Ahmed, Ryan A Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L Willke, and Hoda Eldardiry. 2017. A Framework for Generalizing Graph-based Representation Learning Methods. *arXiv:1709.04596* (2017).

[3] Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: predicting and recommending links in social networks. In *WSDM'11*. ACM, 635–644.

[4] Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, Vol. 14. 585–591.

[5] Kendrick T Brown, Tony N Brown, James S Jackson, Robert M Sellers, and Warde J Manuel. 2003. Teammates on and off the field? Contact with black teammates and the racial attitudes of white student athletes. *J. Appl. Soc. Psychol* 33, 7 (2003).

[6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *AAAI'16*. AAAI Press, 1145–1152.

[7] Albert Cui, Howard Chung, and Nicholas Hanson-Holtry. [n. d.]. YASP 3.5 Million Data Dump. ([n. d.]).

[8] Anders Drachen, Matthew Yancey, John Maguire, Derrek Chu, Iris Yuhui Wang, Tobias Mahlmann, Matthias Schubert, and Diego Klabajan. 2014. Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2). In *Games media entertainment (GEM), 2014 IEEE*. IEEE, 1–8.

[9] Christoph Eggert, Marc Herrlich, Jan Smeddinck, and Rainer Malaka. 2015. Classification of player roles in the team-based multi-player game dota 2. In *International Conference on Entertainment Computing*. Springer, 112–125.

[10] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* (2018). https://doi.org/10.1016/j.knosys.2018.03.022

[11] Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. TrueSkill: a Bayesian skill rating system. In *Advances in neural information processing systems*. 569–576.

[12] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *arXiv:1611.07308* (2016).

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

[14] Alex Leavitt, Brian C Keegan, and Joshua Clark. 2016. Ping to win?: Non-verbal communication and team performance in competitive online multiplayer games. In *Proc. Conference on Human Factors in Computing Systems*. ACM, 4337–4350.

[15] Alexandru Losup, Ruud Van De Bovenkamp, Siqi Shen, Adele Lu Jia, and Fernando Kuipers. 2014. Analyzing implicit social networks in multiplayer online games. *IEEE Internet Computing* 18, 3 (2014), 36–44.

[16] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proc. of ACM SIGKDD*. 1105–1114.

[17] Hyunsoo Park and Kyung-Joong Kim. 2014. Social network analysis of high-level players in multiplayer online battle arena game. In *Social Informatics*. 223–226.

[18] Jone L Pearce. 1993. Toward an organizational behavior of contract laborers: their psychological involvement and effects on employee co-workers. *Academy of Management Journal* 36, 5 (1993), 1082–1096.

[19] Nataliia Pobiedina, Julia Neidhardt, Maria del Carmen Calatrava Moreno, and Hannes Werthner. 2013. Ranking factors of team success. In *WWW*. 1185–1194.

[20] Anna Sapienza, Alessandro Bessi, and Emilio Ferrara. 2018. Non-negative tensor factorization for human behavioral pattern mining in online games. *Information* 9, 3 (2018), 66.

[21] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009), 4.

[22] Tim Urdan and Erin Schoenfelder. 2006. Classroom effects on student motivation: Goal structures, social relationships, and competence beliefs. *Journal of school psychology* 44, 5 (2006), 331–349.

[23] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proc. 22nd SIGKDD*. ACM, 1225–1234.

[24] Pu Yang, Brent E Harrison, and David L Roberts. 2014. Identifying patterns in combat that are predictive of success in MOBA games.. In *FDG*.