


A Recommendation System for People Analytics

Nan Wang, DeepMacro, USA

Evangelos Katsamakos, Gabelli School of Business, Fordham University, USA

 <https://orcid.org/0000-0002-3249-2017>

ABSTRACT

Companies seek to leverage data and people analytics to maximize the business value of their talent. This article proposes a recommendation system for personalized workload assignment in the context of people analytics. The article describes the system, which follows a novel two-level hybrid architecture. The authors evaluate the system performance in a series of computational experiments and discuss future extensions. Overall, the proposed system could create significant business value as a decision support system that could help managers make better decisions. The article demonstrates how computational and machine learning approaches can complement humans in improving the performance of organizations.

KEYWORDS

Collaborative Filtering, Data Science, Decision Support System, Network Analysis, People Analytics, Recommendation System, Software Development, Talent Analytics, Workload Assignment

INTRODUCTION

People analytics is a data-driven approach to managing people at work. Companies use data and analytics to create effective and fair solutions regarding employee recruitment, performance management, promotion, and retention (Allen *et al.* 2010; Davenport *et al.* 2010). Google, for example, is leveraging workplace data to motivate employees and create a high-quality working environment (Bloomberg, 2015). People analytics is sometimes called talent analytics.

This article aims to contribute a new system to the people analytics field. It proposes a novel recommendation system for personalized workload assignment to optimize talent performance and productivity.

Recommendation (recommender) systems have been used successfully in various data science applications and domains (Adomavicius & Tuzhilin, 2005). We believe there is a great promise of using recommendation systems in the context of people analytics as well. This approach is most relevant in organizations with rich data and trackable metrics, as in the case of software development.

This article uses non-publicly available data from a software firm to build a workload assignment recommendation system for people analytics. The article introduces a novel *two-level hybrid* (2LH) recommendation system. 2LH has two base hybrids and integrates them in the upper level: *organizational network analysis* (ONA) based hybrid and graph projection-based hybrid. For base I hybrid, we employ *network analysis* to profile users and identify neighbors. We show that this is an

DOI: 10.4018/IJBIR.20210701.oa4

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

Table 1. Sample of original dataset. Coding Effort measures software developers' contribution to repositories in terms of a series of metrics including volume, complexity and interrelatedness of codes (Newswire 2013).

	Worker ID	Repository ID	Coding Effort	Date
0	aaa	AAA	0.00	2015-07-01
1	dxw	HEL	5.00	2015-07-01
2	woe	WHS	4.73	2015-07-01
3	woe	EZR	0.26	2015-07-01
4	wlk	RJA	5.00	2015-07-01

effective user profiling approach and a solution for sparsity. For base II hybrid, we employ cross-correlation as a complementary similarity metric to a recently proposed one (*two-step random walk*). We discuss standard performance metrics and introduce a customized one: *frequency of the system failing to recommend at least 10 items*. We empirically validate the predictive accuracy and recommendation effectiveness of the proposed approach. The merits of the system include scalability, flexibility, and extensibility. We also discuss future extensions and applications of our recommendation system that could create business value.

The next section describes the dataset and background. We then present the recommendation system and evaluate its performance. The last section discusses the findings and identifies opportunities for future work.

DATA AND BACKGROUND

We describe the dataset and collaborative filtering in recommendation systems.

Dataset

The dataset contains the contribution and efforts of 2621 developers to 1705 repositories in a real-life enterprise in a period of 92 days (from July 1, 2015 to September 30, 2015). Data has four variables, namely Date, Developer ID, Repository ID and Coding Effort.

There are 172,354 records in total, where “Developer ID” and “Repository ID” respectively identify each unique developer and repository. Data regarding developers activity is collected daily, using source code repositories like Subversion and Git, and task tracking systems such as Jira. Table 1 shows a subset from the dataset.

As long as a developer is involved in a repository, coding effort is recorded regardless of the presence or absence of contribution. To keep the information of developers' involvement as a supplement to their coding effort, we retain all data records, even when coding effort is zero.

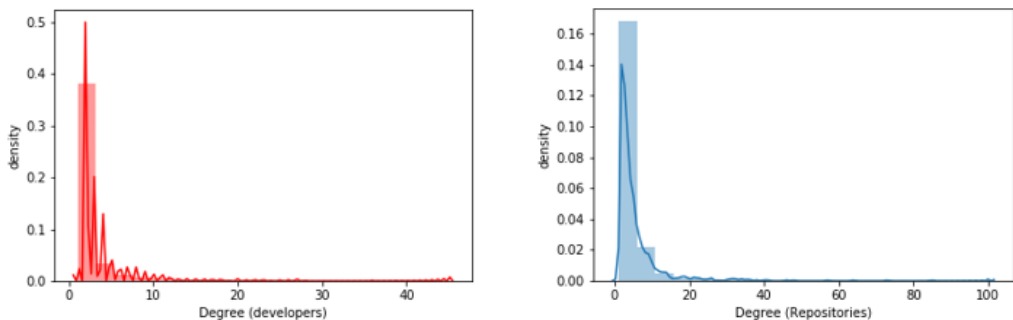
We represent data as a bipartite weighted graph ($G_{dev-rep}$) with developers being one class of nodes and repositories another. Edges are weighted by a developer' total coding effort to a repository. Table 2 shows the attributes of the static graph over the entire time period.

Table 2. Graph attributes

Number of developer nodes	2621
Number of repository nodes	1705
Number of edges	6414
Average weight for edges	60
Average degree of developer nodes	2.45
Average degree of repository nodes	3.76

Figure 1 shows the density of degree centrality (the number of ties that a node has) for developer nodes and repository nodes. Notice that the degree of both types of nodes centers around one. More than half of developers (1440 out of 2621) work in only one repository, and many repositories (725 out of 1705) are individual repositories. Data is sparse under this situation, which constitutes a problem for most collaborative filtering solutions.

Figure 1. Degree distribution for developer nodes and repository nodes



Collaborative Filtering

Collaborative filtering (CF) is a method of making predictions, or recommendations, about the interests of a user by collecting preferences or taste information from many users (collaborative) (Su & Khoshgoftaar 2009).

There are three major types of CF algorithms. Memory-based approaches use user rating data to compute the similarity between users or items. Model-based approaches use machine learning algorithms to predict users' rating of unrated items. Hybrid approaches combine memory-based and model-based algorithms to avoid certain limitations of memory-based and model-based models ((Adomavicius & Tuzhilin 2005; Ricci et al. 2011).

SYSTEM DESCRIPTION

We propose a *two-level hybrid* (2LH) recommendation system for personalized work assignment in people analytics. 2LH has two base hybrids: ONA based hybrid and graph projection based hybrid. The two base hybrids are integrated in the upper level of the system.

First, we discuss rating normalization and then describe in detail the system components.

Rating Normalization

In movie or music recommendation systems, ratings can be largely affected by diversity in users' expression of preferences. For example, user i and user j both like movie A . However, user i might give the movie 4.9 on a scale of 5 while user j only gives 4.5, because user i thinks only a rating as high as 4.9 can show his or her strong preference, while user j feels that a rating of 4.5 is high enough.

To overcome this problem of scaling diversity, ratings are normalized. In our context, we consider coding effort as a "rating" measuring developer's "preference" to a repository, and thus we normalize it.

As a normalization method, we subtract the coding effort of a developer from her average coding effort and add a scaling constant to make all coding effort positive (1).

$$\widetilde{r}_{i,b} = r_{i,b} - \bar{r}_i + \beta$$

Base Hybrid I: Organizational Network Analysis Based

Our first hybrid model (*Base hybrid I*) incorporates model-based characteristics into a memory-based approach, to which we apply *organizational network analysis* (ONA). ONA can provide a comprehensive view of an organization regarding information flow and collaboration. We identify neighbors by grouping users with community detection techniques (Fortunato 2010; Malliaros & Vazirgiannis 2013; Porter et al. 2009) and clustering on user profiles.

Our *Base hybrid I* modeling process has five steps: (1) User Profiling for Clustering; (2) Find Neighbors through Clustering; (3) Find Neighbors through Community Detection; (4) Assign Recommendation Power to Neighbors; (5) Recommend Repositories.

User Profiling for Clustering

First, we project the *developer-repository graph* ($G_{dev-rep}$) into one-mode *developer-developer graph* ($G_{dev-dev}$) in a binary way. Edge weights are not considered.

Then, we extract graph property metrics, such as degree centrality, betweenness centrality, closeness centrality, eigenvector centrality, and PageRank as profiling features (Newman 2001). These metrics are used to infer developers' roles and influence (Rodriguez et al. 2010). For example, nodes having high PageRank tend to be *central people* who play prominent roles in organizations, and nodes carrying more significant betweenness centrality are influencers in information sharing.

Find Neighbors through Clustering

We apply four *Gaussian mixture models* with varied covariance structures (Pedregosa et al. 2011) to group nodes together based on generated user profiles. The cluster number is optimized using Bayesian information criterion (BIC) (Bhat and Kumar 2010).

Find Neighbors through Community Detection

Community detection provides insights about the overall network structure, behavioral patterns of nodes, and their relations. In our case, community detection reveals insights about developers' collaboration preference.

We apply four commonly-used and well-performed community detection algorithms, namely *Fast Greedy*, *Walktrap* (Pons & Latapy 2006), *Infomap*, and *Louvain* (Blondel et al. 2008; Csardi & Nepusz 2006).

The motivation of using both clustering and community detection is to better group developers with a thorough consideration of both their connections and attributes. Rather than limited to one single cluster, developers can belong to several different subgroups.

Assign Recommendation Power to Neighbors

As we apply four Gaussian mixture models and four community detection models to identify neighbors, each developer will have neighbors from eight sources. Therefore, we define a weighting scheme to assign neighbors' *recommendation power*. *Recommendation power* (RP) indicates how much a user is willing to let others represent his or her preference (Zhou et al. 2007).

A subgroup-number based weighting scheme (SNBS) is applied. Inspired by inverse user frequency (Breese et al. 1998), we define $\log \frac{C_n}{\beta}$ as recommendation power of neighbors identified by n^{th} model (RP_n). C_n is the number of subgroups generated by n^{th} model, and β is a scaling parameter. β is greater than one and determined by grid search (Bergstra and Bengio 2012).

$$RP_n = \log \frac{C_n}{\beta}$$

Suppose that model A divides all developers into 50 subgroups, while model B divides them into 200 subgroups. Neighbors identified by A will have RP of $\frac{50}{\beta}$, while those by B will have $\log \frac{200}{\beta}$. The intuitive way to interpret SNBS is that as B divides developers into more subgroups, it differentiates them in a more detailed way.

We will compare model performance with and without SNBS to validate the effectiveness of SNBS.

Recommend Repositories

We then recommend repositories in a descending order of their recommendation scores. (3) shows the scoring function for repository b recommended to developer i .

$$RS_{b,i} = \sum_{n=1}^8 \sum_{j_n \in J} (W_{b,j_n} \times RP_n)$$

Base Hybrid II: Graph Projection Based

The second hybrid model implements several memory-based models and merges the results with a voting scheme. Memory-based models are in the form of graph projection, with weight allocation being the equivalent of similarity calculation. $w_{i,j}$ is the weight of edge between i and j on the projected developer-developer graph ($G_{dev-dev}$). This weight is equal to the similarity (*sim*) of i and j .

There are three steps to building the projection hybrid: (1) Calculate Neighbor Similarity with Four Projection Methods; (2) Generate Four Recommendation Lists; (3) Merge Recommendation Lists.

Calculate Neighbor Similarity with Four Projection Methods

The four weight allocation methods we apply are: *Two-step random walk* (Shang et al. 2008), *Pearson's coefficient*, *cross-correlation similarity* (Papoulis 1962; Bracewell 1965), and *Manhattan distance*.

Two-step random walk process has been proved superior as a weight allocation method. (4) calculates the edge weight between node i and node j on the projected graph based on this metric.

$$w_{i,j} = \sum_{a \in A} \left(\frac{W_{i,a}}{W_i} \times \frac{W_{a,j}}{W_a} \right)$$

Pearson's correlation coefficient (ρ) is the covariance of the two variables divided by the product of their standard deviations. If we use $\rho(i, j)$ to represent Pearson's correlation coefficient between developer i and j , the weight of edge between node i and j on the projected graph is $\rho(i, j)$.

The intuition for both *Pearson's coefficient* and *cross-correlation similarity* is that if two developers behave similarly in shared repositories in terms of coding contribution, they might share skillsets or proficiency level. Thus they are similar and have a higher chance to collaborate in the future.

Cross correlation has wide application in the fields of pattern recognition, but hasn't been popular as similarity metric in collaborative filtering.

In the above analysis, we assume developer similarity as an explanation for high collaboration frequency. However, developers having diverse skills are more likely to collaborate. Repositories need mixed expertise (testers, reviewers, supervisors, etc.), and developers showing different working patterns are complementary to each other.

$$w_{i,j} = MD_{i,j} = \sum_{b \in B} |W_{b,i} - W_{b,j}|$$

Generate Four Recommendation Lists

The recommendation score function for a repository is the sum of the product of corresponding edge weights on $G_{dev-rep}$ and $G_{dev-dev}$ (6). Repositories are recommended in descending order of this score. As we apply four projection methods, we will get four recommendation lists.

$$RS_{b,i} = \sum_{j \in J} (W_{b,j} \times w_{i,j})$$

Merge Recommendation Lists

We use a customized voting scheme to merge results. Assume for each developer, model M generates a recommendation list of $\{M_1, M_2, M_3, M_4, \dots, M_n\}$, while model G gives $\{G_1, G_2, G_3, G_4, \dots, G_n\}$. If model M performs better than G in terms of $MAP@10$, our ensemble results would be $\{M_1, G_1, M_2, G_2, \dots, M_n, G_n\}$. After de-duplicating the list, we take the first ten items as our final recommendation list.

Second-Level Hybrid

The second-level hybrid model is created by merging predictions from two base hybrids with the same voting scheme described earlier. In particular, suppose base I model generates a list of $\{I_1, I_2, I_3, I_4, \dots, I_n\}$ while base II model generates a list of $\{II_1, II_2, II_3, II_4, \dots, II_n\}$. If base I model is better than base II in terms of $MAP@10$, then our final recommendation list is $\{I_1, II_1, I_2, II_2, I_3, II_3, I_4, II_4, \dots\}$.

PERFORMANCE EVALUATION

First, we discuss the training-testing split of the data, then performance metrics and experimental results.

Training and Testing Split

We split the data into the training set and testing set, with the former to construct graphs, identify neighbors, build models and generate recommendations, and the latter to validate and evaluate model performance.

As mentioned before, the data covers 92 days of developers' activities, from July 1 to September 30. We split the training and testing set following the order of time. Typical ratios for training and testing split are like 80% versus 20%, or 75% versus 25%, with the training set containing many more records than the testing set.

Our modeling process doesn't require massive training data to reach accurate results. Furthermore, most working routines, take one or more weeks. Thus, we use three training-test split schemes: 7–85, 14–78, 21–71, which respectively use first 7 days, 14 days and 21 days of developers' behavior as training data.

Performance Metrics

Traditional metrics for evaluating the performance of recommender systems, for example, *precision*, *recall*, and *ROC curve*, are focused on two aspects: coverage and accuracy (Adomavicius & Tuzhilin 2005). Advanced metrics, such as *mean average precision* and *discounted cumulative gain*, take the order of retrieved documents into account (Jarvelin & Kekalainen 2002).

We take accuracy, coverage and order into consideration. Applied metrics are *mean average precision at K* ($MAP@K$) (Manning et al. 2008) and *recall @ K* ($Recall@K$).

We set K to be 10. In other words, we give each developer ten suggested repositories. Then based on whether they work on our recommendations in the testing periods, we evaluate the predictive accuracy of our model.

Furthermore, due to data sparsity, systems may generate no or few recommendations for some users. Users' choices are accordingly limited, and the effectiveness of the system is diminished. To ensure that our system generates at least ten options for as many developers as possible, we propose *frequency of recommendation lists shorter than 10 items* ($R_{l<10}$) as another metric. $R_{l<10}$ is zero when our system recommends ten repositories to all developers, which is the ideal case.

Experiments and Results

We evaluate and compare 2LH model with seven other models in terms of three metrics $MAP@10$, $Recall@10$ and $R_{l<10}$ under three training-test split schemes (7–85, 14–78, 21–71). Tables 3 to 8 show the results of the experiments.

The seven other models used for comparison are unweighted *RP* hybrid (base I without SNBS), weighted *RP* hybrid (base I), *two-step random walk graph projection* (2step-P), *cross-correlation*

Table 3. Descriptions of training and testing graph

	Total nodes	Total Edges	Avg. number of edges	Avg. number of new repositories	Number of developer to recommend
Train set	2321	3172	1.37	-	-
Test set	2775	6228	2.24	2.34	824

graph projection (CC-P), Pearson's coefficient graph projection (Pearson-P), Manhattan-distance graph projection (MD-P) and projection-based hybrid (base II).

7 – 85 Training-test split

14 – 78 Training-test split:

21 – 71 Training-test split:

The results show that our proposed approach, 2LH, is the best model since, under three training-testing split schemes, it consistently outperforms all other models in terms of all three metrics. The 2LH approach manages to absorb the merits of two base hybrids and counterbalance their defects.

Moreover, all eight models can provide recommendations to all targeted developers. Two base hybrids overall outperform corresponding individual models. Especially for base I hybrid, both and are improved compared to unweighted hybrid. This result validates the effectiveness of the proposed weighting scheme (SNBS).

Projection-based models (2step-P, CC-P, Pearson-P, MD-P and base II) universally have high $MAP@10$, and CC-P and 2step-P methods have the highest values. However, they all suffer from high $R_{l<10}$ due to data sparsity. For example, under 7 – 85 Training-test split, $R_{l<10}$ of projection methods is as high as 97.7%, meaning that 97.7% of developers are unable to receive at least ten recommendations. In contrast, ONA based models (unweighted RP hybrid and base I) perform poorly regarding $MAP@10$ but achieve good $R_{l<10}$.

DISCUSSION

Companies use people analytics technology to maximize the value of their talent as they seek to compete effectively and gain competitive advantage. This article proposed a novel recommendation

Table 4. Model results

	unweighted <i>RP</i> hybrid	2step-P	CC-P	Pearson-P	MD-P	base I	base II	2LH
$MAP@10$	0.2044	0.3698	0.3675	0.3416	0.3363	0.3279	0.3689	0.3787
$Recall@10$	0.3446	0.4911	0.4890	0.4796	0.4865	0.4786	0.4906	0.4970
$R_{l<10}$	9.0%	97.7%	97.7%	97.7%	97.7%	9.0%	97.7%	8.7%

Table 5. Descriptions of training and testing graph

	Total nodes	T o t a l edges	Avg. number of edges	Avg. number of new repositories	Number of developer to recommend
Train set	2350	3635	1.55	-	-
Test set	2577	6018	2.34	2.31	734

Table 6. Model results

	unweighted <i>RP</i> hybrid	2step-P	CC-P	Pearson-P	MD-P	base I	base II	2LH
<i>MAP@10</i>	0.1956	0.3645	0.3674	0.3389	0.3366	0.3222	0.3666	0.3782
<i>Recall@10</i>	0.3391	0.5129	0.5129	0.5025	0.5117	0.5105	0.5165	0.5497
$R_{l<10}$	3.7%	85.1%	85.1%	85.1%	85.1%	3.7%	85.1%	3.4%

Table 7. Descriptions of training and testing graph

	Total nodes	Total edges	Avg. number of edges	Avg. number of new repositories	Number of developer to recommend
Train set	2377	3986	1.68	-	-
Test set	2547	5830	2.29	2.26	678

Table 8. Model results

	unweighted <i>RP</i> hybrid	2step-P	CC-P	Pearson-P	MD-P	base I	base II	2LH
<i>MAP@10</i>	0.1748	0.3593	0.3608	0.3277	0.3355	0.2942	0.3615	0.3673
<i>Recall@10</i>	0.3307	0.5375	0.5297	0.5154	0.5272	0.4953	0.5390	0.5568
$R_{l<10}$	2.5%	79.6%	79.6%	79.6%	79.6%	2.5%	79.6%	2.5%

system for personalized workload assignment to help managers make better decisions in the context of people analytics.

In this work, we see people analytics as a systems thinking challenge that can be solved computationally using a recommendation system approach. The approach captures system complexity in the form of graph structure, graph metrics, and graph algorithms.

The high-level architecture of our system is described as a two-level hybrid (2LH). The proposed system outperformed other methods in terms of predictive accuracy and recommendation effectiveness. The proposed recommendation system is also promising in terms of its scalability, flexibility, and extensibility. As only the past few weeks' data is used to generate recommendations, the model is computationally efficient and thus scalable. Overall, our evaluation through a series of computational experiments suggests that such a system could add value in the context of people analytics initiatives in companies (Wang & Katsamakas 2019).

The proposed workload assignment recommendation system can also be seen as a *Decision Support System* (DSS). Companies would benefit the most by using the system as a tool aiding and supporting management decisions. Despite the renewed fears about technology replacing humans, we believe that most of the value of machine-learning techniques, like the proposed system, comes from augmenting and supporting humans in making better decisions.

Future extensions of the proposed system could further enhance system performance. For example, considering the influence of distant nodes can effectively improve performance (Palau et al 2004). Another extension is to employ more advanced clustering techniques in ONA based hybrid. Future work could also consider contextual information (Adomavicius & Tuzhilin 2011), such as workload schedule, budget, and business objectives. Other work could incorporate economics-oriented metrics that capture the business value of recommendations.

REFERENCES

- Adomavicius, G., & Tuzhilin, A. (2005). Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749. doi:10.1109/TKDE.2005.99
- Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In *Recommender Systems Handbook*. Springer.
- Allen, D. G., Bryant, P. C., & Vardaman, J. M. (2010). Retaining talent: Replacing misconceptions with evidence-based strategies. *The Academy of Management Perspectives*, 24(2), 48–64.
- Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 281–305.
- Bhat, H. S., & Kumar, N. (2010). *On the derivation of the Bayesian Information Criterion*. Academic Press.
- Blondel Vincent, D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). *Fast unfolding of communities in large networks*. arXiv:0803.0476.
- Bloomberg. (2015). *Google's Using Workplace Data to Build a Better Employee*. Retrieved from <https://www.bloomberg.com/news/videos/2015-11-11/google-s-using-workplace-data-to-build-a-better-employee>
- Bracewell, R. (1965). Pentagram Notation for Cross Correlation. *The Fourier Transform and Its Applications*. New York: McGraw-Hill.
- Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*.
- Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research, *InterJournal. Complex Systems*, 1695. <http://igraph.org>
- Davenport, T. H., Harris, J., & Shapiro, J. (2010). Competing on talent analytics. *Harvard Business Review*, 88(10), 52–58. PMID:20929194
- Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3–5), 75–174. doi:10.1016/j.physrep.2009.11.002
- Jarvelin, K., & Kekalainen, J. (2004). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4), 422–446. doi:10.1145/582415.582418
- Jones, E., Oliphant, E., & Peterson, P. (2001). *SciPy: Open Source Scientific Tools for Python*. <https://www.scipy.org/>
- Konstan, J. (2014). Teaching Recommender Systems at Large Scale: Evaluation and Lessons Learned from a Hybrid MOOC. *ACM Transactions on Computer-Human Interaction*, 22, 61–70.
- Malliaros, F. D., & Vazirgiannis, M. (2013). Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4), 95–142. doi:10.1016/j.physrep.2013.08.002
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. doi:10.1017/CBO9780511809071
- Newman, M. E. J. (2001). Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1), 016132. doi:10.1103/PhysRevE.64.016132 PMID:11461356
- Newswire. (2013). *BlueOptima Coding Effort Analytics™ Analytics Announces Support for Git*. Retrieved from <https://www.newswire.com/blueoptima-coding-effort-analytics/252989>
- Palau, J. (2004). Collaboration Analysis in Recommender Systems Using Social Networks. In M. Klusch, S. Ossowski, V. Kashyap, & R. Unland (Eds.), *Lecture Notes in Computer Science: Vol. 3191. Cooperative Information Agents VIII. CIA 2004*. Springer. doi:10.1007/978-3-540-30104-2_11
- Papoulis, A. (1962). *The Fourier Integral and Its Applications*. New York: McGraw-Hill.
- Pedregosa, . (2011). Scikit-learn: Machine Learning in Python. *JMLR*, 12, 2825–2830.

- Pons, P., & Latapy, M. (2006). Computing Communities in Large Networks Using Random Walks. *International Symposium on Computer and Information Sciences*, 284–293. doi:10.7155/jgaa.00124
- Porter, M. A., Onnela, J.-P., & Mucha, P. J. (2009). Communities in Networks. *Math. Soc.*, 56, 1082–1097, 1164–1166.
- Ricci, F., & Rokach, L., & Shapira, B. (2011). *Introduction. In Recommender Systems Handbook*. Springer.
- Rodriguez, M. G., Leskovec, J., & Krause, A. (2010). Inferring networks of diffusion and influence. *KDD: Proceedings / International Conference on Knowledge Discovery & Data Mining. International Conference on Knowledge Discovery & Data Mining*, 10.
- Shang, M., Fu, Y., & Chen, D. (2008). Personal Recommendation Using Weighted BiPartite Graph Projection. *Apperceiving Computing and Intelligence Analysis, 2008. ICACIA 2008. International Conference*.
- Shirkhorshidi, A. (2015). *A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data*. 10.1371/journal.pone.0144059
- Su, X. Y., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence, 2009*, 1–19. doi:10.1155/2009/421425
- Wang, N., & Katsamakos, E. (2019). A Network Data Science Approach to People Analytics. *Information Resources Management Journal*, 32(2), 28–51. doi:10.4018/IRMJ.2019040102
- Yang, Z., Cheng, H., & Yu, J. X. (2009). Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, 2, 1.
- Zhang, Y., Callan, J., & Minka, T. (2002). Novelty and Redundancy Detection in Adaptive Filtering. *Proc. 25th Ann. Int'l ACM SIGIR Conf.*, 81-88. doi:10.1145/564376.564393
- Zhou, T., Jiang, L.L., Su, R. Q., & Zhang, Y.C. (2007). *Effect of initial configuration on network-based recommendation*. arXiv:0711.2506.
- Zhou, T., Ren, J., Matus, M., & Zhang, Y.-C. (2007). Bipartite network projection and personal recommendation. *Physical Review. E*, 76(4), 046115. doi:10.1103/PhysRevE.76.046115 PMID:17995068

Nan Wang is a data scientist at DeepMacro, a NYC based macro investing company. Her work involves supporting systematic investment strategies in US and emerging markets, through looking at Macro Economics in a data-driven way. The data she deals with on a daily basis is a combination of structured (e.g. time series) and unstructured (e.g. large images), to which she applies statistical exploratory analysis and machine learning modeling. She has strong research interest in the field of social network analysis and applied machine learning. She holds an MS degree in Business Analytics from Gabelli School of Business, Fordham University, where she also worked as research assistant.

Evangelos “Evan” Katsamakos is Professor of Information, Technology & Operations, at Gabelli School of Business, Fordham University. He is a faculty member since 2004. He served as department chair from July 2012 until June 2018 leading the growth of curriculum, enrollments and faculty positions. Professor Katsamakos’ research analyzes the strategic and economic impact of digital technologies focusing on digital transformation, platform strategies, data science, business analytics, and fintech. His research interests include economic theory and analytical modeling, machine learning and computational modeling of complex business systems. Prof. Katsamakos’ research has appeared in *Management Science*, *Journal of Management Information Systems*, *System Dynamics Review*, *International Journal of Medical Informatics*, *Information Resources Management Journal*, *Electronic Commerce Research and Applications*, *Business Process Management Journal* and in multiple other scholarly journals, conference proceedings and books. His research on digital innovation received the 2016 SIM (Society of Information Management) Best Academic Paper Award. Prof Katsamakos is an Associate Editor of the *European Journal of IS*, among other journals. He served as guest editor of the special issue on the *Dynamics of Information Systems* published in Fall 2008 at the *System Dynamics Review*. Prof. Katsamakos has taught graduate and undergraduate business school courses including *Business Tech & Analytics*, *Business Analytics Integrated Project*, *Data Mining*, *Text Analytics*, *Systems Analysis and Design*, *E-business Strategies and Applications*, *Cloud Computing*, and *Tech Startups*. He received the 2018 Dean’s Award for Teaching Innovation for his contribution to curriculum innovation. He has a strong interest in Fintech innovation: he participated in several conferences, supervised student projects, advised the student Fintech club and designed, in collaboration with Finance area, a Fintech concentration and Fintech courses. Professor Katsamakos holds a Ph.D. from the Stern School of Business, New York University, an M.Sc. from the London School of Economics and a Computer Science and Engineering degree from the University of Patras, Greece.