Intelligent Agents with Graph Mining for Link Prediction over Neo4j

Michalis Nikolaou[1], Georgios Drakopoulos[2] [a], Phivos Mylonas[3] [b] and Spyros Sioutas[1] [c]

[1]*Computer Engineering and Informatics Department, University of Patras, Patras, Greece*
[2]*Department of Informatics, Ionian Univerity, Kerkyra, Greece*
[3]*Informatics and Computer Engineering Department, University of West Attica, Athens, Greece*

Keywords: Intelligent Agents, Network Structural Integrity, Connectivity Patterns, Link Prediction, Graph Mining, Neo4j.

Abstract: Intelligent agents (IAs) are highly autonomous software applications designed for performing tasks in a broad spectrum of virtual environments by circulating freely around them, possibly in numerous copies, and taking actions as needed, therefore increasing human digital awareness. Consequently, IAs are indispensable for large scale digital infrastructure across fields so diverse as logistics and long supply chains, smart cities, enterprise and Industry 4.0 settings, and Web services. In order to achieve their objectives, frequently IAs rely on machine learning algorithms. One such prime example, which lies in the general direction of evaluating the network structure integrity, is link prediction, which depending on the context may denote growth potential or a malfunction. IAs employing machine learning algorithms and local structural graph attributes pertaining to connectivity patterns are presented. Their performance is evaluated with metrics including the F1 score and the ROC curve on a benchmark dataset of scientific citations provided by Neo4j containing ground truth.

1 INTRODUCTION

Intelligent agents (IAs) are advanced and flexible autonomous piece of software tailored for pursuing multiple and potntially conflicting objectives whose ranking may change over time while operating in an evolving environment which may not be fully known (Li et al., 2022a). In this sense IAs are invaluable for supporting digital awareness extending the human prescient ability with regards to digital infrastructure.

In order for IAs to be effective in their respective missions, they are frequently augmented with machine learning (ML) capabilities. For instance, it would make sense to use ML to enhance communication reliability in an industrial or a smart city context as communication channels are in practice constantly subject to various sources of noise depending on the operational setting. This holds especially true in distributed systems where wired links are corrupted by white Gaussian noise, while mobile links are prone to lognormal noise, shadowing, or even intersymbol interference (ISI) from any nearby links.

This has been further enhanced with the advent of graph signal processing (GSP), a field which has recently garnered intense interdisciplinary research attention as it allows for established ML techniques to be applied on a graph mining setting. IAs can benefit from GSP since most of the infrastructure, whether digital or physical, they routinely operate on is or at least can be represented by a graph. Common examples include long supply chains, power and water networks, and computer networks.

The primary research objective of this conference paper is the development of an IA capable of performing link prediction based on a wide array of ML algorithms running on local attributes such as the Adamic-Adar index and the resource allocation metric. Since numerous local decisions are critical for the emergence of global patterns in scale free graphs, the effect of the former can be evaluated on a macroscopic scale through metrics like the F1 score and the receiver operating characteristic (ROC). This work differentiates itself from existing approaches by heavily focusing on graph locality properties. The benchmark dataset is represented in a Neo4j database and in fact it was taken directly from the Neo4j developer site[1].

The remaining of this work is structured as follows. In section 2 the recent scientific literature re-

[a] https://orcid.org/0000-0002-0975-1877
[b] https://orcid.org/0000-0002-6916-3129
[c] https://orcid.org/0000-0003-1825-5565

---

[1]https://neo4j.com/developer/graph-data-science/link-prediction/graph-data-science-library/

garding IAs, GSP, and graph mining strategies is briefly overviewed. In section 3 the IA architecture and its functionality are explained. The results obtained by the proposed methodology are analysed in section 4. Future research directions are given in section 5. Bold capital letters denote matrices, small boldface vectors, and small plain scalars. Function parameters come with a semicolon after their arguments. Technical acronyms are explained the first time they are encountered in the text. Finally, notation is summarized in table 1.

2 PREVIOUS WORK

A major part of IA functionality (Moussawi et al., 2023) is the interaction with their environment (Chiatti et al., 2022), especially for enhancing human digital awareness (Riedl, 2019). IA design elements influence among others user acceptance (Elshan et al., 2022), seamless integration with various ML algorithms (Kaswan et al., 2022), presentation to the final user (Narumi et al., 2022), coordination with other IAs (Nefla et al., 2022), and Web mining optimization (Yoon et al., 2022). IA applications include strategically searching LinkedIn for trusted candidates (Drakopoulos et al., 2020c), monitoring long food supply chains (Pirsa et al., 2022), and digital health (Chen et al., 2022). ML algorithms which have been used by IAs are shown in table 2.

GSP is an emerging field where graphs are two-dimensional signals from an irregular domain (Ortega, 2022) which can be combined with graph neural networks (GNNs) (Li et al., 2022a) or other ML techniques (Shi and Moura, 2022) to extract knowledge from graphs. Applications include cultural content recommendation based on affective attributes (Drakopoulos et al., 2022), collaborative filtering (Liu et al., 2023), the smart decompression of long graph sequences (Drakopoulos et al., 2021a), graph reconstruction with Sobolev smoothness (Giraldo et al., 2022), GNNs for evaluating the affective coherence of fuzzy (Drakopoulos et al., 2021b) and ordinary (Drakopoulos et al., 2020a) Twitter graphs, transfer learning between graphs (Ruiz et al., 2023), aproximating directed graphs with undirected ones using matrix polar factorization (Drakopoulos et al., 2021c), ensemble learning (Shang et al., 2022), intelligent fault diagnosis (Li et al., 2022b), probabilistic approximation of topological correlation (Drakopoulos and Kafeza, 2020), multiscale learning (Zhao et al., 2022), wide learning of massive graphs (Gao et al., 2022), and community discovery on Twitter based on multiple criteria (Drakopoulos et al., 2020b).

Graph mining is a broad field (Dong et al., 2023) encompassing quite diverse research such as opinion mining (Lin et al., 2022), enhanced ontologies (Drakopoulos et al., 2017), parallel graph processing (Dai et al., 2022), face recognition (Bedre and Prasanna, 2022), heuristic community discovery (Drakopoulos and Mylonas, 2022), and finding interrelated requirements in software architectures from large texts (Singh, 2022).

3 INTELLIGENT AGENT DESIGN

The conceptual architecture of an IA is shown at figure 1. Notice how any IA can interact besides the digital realm also with its physical environment by accepting input, even incomplete or fuzzy depending on the operational context, through a broad array of available heterogeneous sensors depending on its configuration such as light, atmospheric pressure, humidity, acceleration, and electromagnetic field sensor to name only a few. Therefore IAs can bridge the gap between the physical and the digital realms.

According to the original Russel-Norvig classification there are the following types of IAs:

- **Simple Reflex Agents:** They react to external stimuli but do not take into consideration any history.

- **Model Based Reflex Agents:** They live in partially observable environments and have state history.

- **Goal Based Agents:** They select among a set of desirable goal states and make a course of actions.

- **Utility Based Agents:** They can quantify how desirable a given state is and choose between them.

- **Learning Agents:** They adapt and can dynamically select actions leading to a desired goal.

The proposed IA clearly belongs to the fifth and most general level of the above taxonomy. In semi-supervised and unsupervised learning settings the concept of state is paramount, since it codifies the knowledge IA has for the environment and its past interactions with it. Maintaining such information is vital for an IA operating in the Web since the latter is stateless by dsign. The state vector $\mathbf{s}$ as shown in equation (1) is a column vector containing p state variables which essentially reflect the configuration of the proposed IA and they control its actions. In this particular context, the p variables are the attributes collected for each vertex shown later in this section, suggesting thus that the IA trajectory and ac-

Table 1: Notation of this work.

Symbol	Meaning	First in
$\triangleq$	Definition or equality by definition	Eq. (1)
$\{s_1, \ldots, s_n\}$	Set with elements $s_1, \ldots, s_n$	Eq. (2)
$|S|$	Set cardinality functional	Eq. (2)
(u, v)	Undirected edge between u and v	Algo. 1
$\deg(u)$	Degree of vertex u	Eq. (7)
$\Gamma(u)$	Neighborhood of vertex u	Eq. (6)
$\text{prob}\{\Omega\}$	Probability of event Ω occurring	Eq. (14)

Table 2: Neural network architectures.

| Model	Unsupervised	Supervised	Software
Autoencoder	Yes	No	keras, H2O
Convolutional Deep Belief Network	Yes	Yes	tensorflow, keras, H2O
Convolutional Neural Network	Yes	Yes	tensorflow, fastai
Deep belief network	Yes	Yes	theano, pytorch, tensorflow
Deep Boltzmann machine	No	Yes	boltzmann-machines, pydbm
Denoising autoencoder	Yes	No	tensorflow, keras
Long short-term memory	No	Yes	keras, lasagne, BigDL, Caffee
Multilayer perception	No	Yes	keras, sklearn, tensorflow
Recurrent neural network	No	Yes	keras
Restricted Boltzmann machine	Yes	Yes	pydbm, pylearn2, theanoLM
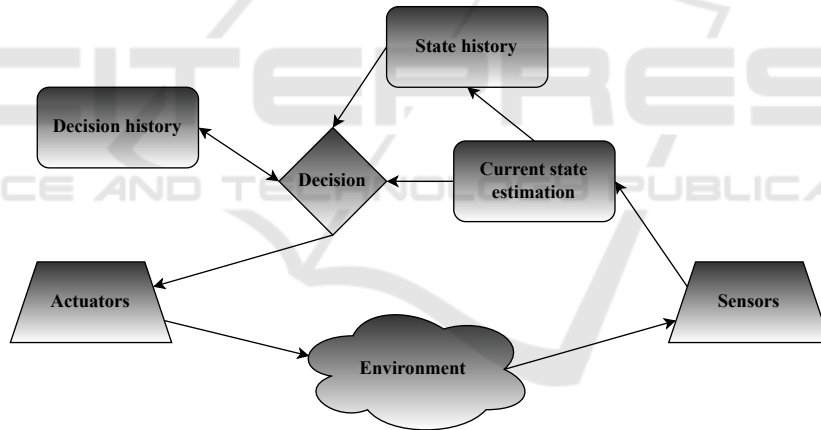
Figure 1: IA architecture.

tions along a given graph rely on vector embeddings.

$$\mathbf{s} \triangleq \begin{bmatrix} h_1 & \ldots & h_p \end{bmatrix}^T \tag{1}$$

The strategy of the proposed IA is outlined in section 1. Its main loop ensures that each vertex is visited exactly once and that the local attribute vector is taken into account. In many IA problems vertex visit strategy is critical, but here a single visit suffices.

The predetermined set of actions can be represented as a set I_a as in (2). Notice that in general depending on the current vertex and the agent state not every action from the set I_a may be applicable or even available. However, here only two actions are available, namely collect vertex attributes and decide

whether a latent edge exists between the local edge and one of its non-neighboring vertices.

$$I_a \triangleq \{a_1, \ldots, a_n\}, \qquad |I_a| = n \tag{2}$$

For each action of (2) there is an associated positive cost, which in general depends on the functional parameters of the IA and the environment, as well as a related reward. In general both can depend on time, but here this is omitted. Action costs and rewards are represented also by two sets as in (3). Observe that I_a, I_c, and I_r have the same cardinality.

$$I_c \triangleq \{c_1, \ldots, c_n\} \quad \text{and} \quad I_r \triangleq \{r_1, \ldots, r_n\} \tag{3}$$

Data: Input graph and IA parameters
Data: Set of termination conditions τ
Result: The predicted edges marked as such
start from a random vertex;
while τ *is not satisfied* **do**
    collect local attributes of u;
    foreach *not connected vertex v* **do**
        collect local attributes of v;
        create joint attributes of u and v;
        if *ML finds a latent* (u,v) **then**
            connect u and v with (u,v);
            mark the above edge as new;
        end
    end
end

Algorithm 1: Proposed IA operation.

The most common figure of merit is the weighted expected cost of the actions taken by the IA as shown in (4). The exponentially forgetting factor γ_0 essentially stipulates that the effect of past errors tend to have a negligible effect on later steps.

$$Q_e \triangleq \sum_{k=1}^{q} \gamma_0^k c_k \qquad (4)$$

An alternative metric is the expected weighted reward to cost ratio of IA as shown in equation (5).

$$Q_r \triangleq \sum_{k=1}^{q} \gamma_0^k \frac{r_k}{c_k} \qquad (5)$$

Frequently the internal mechanics of an IA are modeled as a finite state automaton (FSA) where the transition between each possible state of the state vector **s** depends on the current vertex as well as on the cost and rewards of the available actions. In this context, the capacity and transitivity closure of this FSA can be used as evaluation metrics of the complexity of the underlying IA. Moreover, under the appropriate conditions they may reflect the coding effort necessary for creating the IA. The latter does not have to be mandatorily implemented as an FSA, it only suffices that its internal logic can be translated to one while their implementation is determined by technological levels and the intended IA functionality. For instance, an IA may well be implemented as a rule-based system for AI explainability and accountability purposes.

As FSAs are limited by bounded rationality phenomena, a single IA may not be sufficient for certain purposes, for example considerably suppressing or totally preventing the functionality of a hostile IA. Such scenarios typically require the cooperation of many IAs for creating a distributed hive AI and are commonly modeled as games on graphs.

In this scenario, the performance metric of equation (5) is implicitly used with umiform unit costs for either missing a latent edge between two vertices or declaring that a non-existent edge exists.

3.1 Attributes

In order to train the algorithns determiming the actions and decisions of the proposed IA, a number of graph structural metrics concerning vertex pairs are used. These are also described in the GDSL Web site of Neo4j[2], namely its graph algorithms library.

- The common neighbors score $h_c(u,v)$ stemming from the fundamental property of edge locality, meaning the more neighbors two vertices share, the more probable is that they are connected.

$$h_c(u,v) \triangleq |\Gamma(u) \cap \Gamma(v)| \qquad (6)$$

- The preferential attachment score $h_p(u,v)$ relies on the densification property of dynamic graphs stating that the higher degree a vertex has, the more probable it is to attract new neighbors.

$$h_p(u,v) \triangleq \deg(u)\deg(v) \qquad (7)$$

- The number of total neighbors $h_n(u,v)$ is also derived from the above stated property but takes a different approach by evaluating the joint potential of a vertex pair to attract new neighbors.

$$h_n(u,v) \triangleq \deg(u) + \deg(v) \qquad (8)$$

- The Adamic-Adar index $h_a(u,v)$ evaluates the information theoretic potential for connections of the common graph segment between the two vertices instead of focusing to the vertices.

$$h_a(u,v) \triangleq \sum_{s \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log|\Gamma(s)|} \qquad (9)$$

- The resource allocation metric $h_r(u,v)$ moves along the similar line of reasoning and approximates the reciprocal of the number of connections necessary to be established between them.

$$h_r(u,v) \triangleq \sum_{s \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(s)|} \qquad (10)$$

The attributes are collected using Cypher, the ASCII art based query language of Neo4j which allows access to every graph structural element. The queries necessary for the IA to perform its task.

---

[2]https://neo4j.com/docs/graph-data-science/current/algorithms/linkprediction/

Instead of providing the learning algorithms with the above attributes, they were given an averaged version of them. Since different vertices have different degrees, the harmonic mean of (11) is used:

$$h(u,v) \triangleq \frac{\deg(u)}{\sum_{s \in \Gamma(u)} \frac{1}{h(s,v)}} \qquad (11)$$

The approach of (11) relies heavily on the locality property of the graphs. In other words, it exploits the fact that if v is connected to u, then it is probable that it will be also reconnected to some of its neighbors.

3.2 Learning Algorithms

The algorithms employed by the IA are the following. Notice they are common and readily available in a broad spectrum of platforms.

- Random Forest (RF).
- Decision Tree (DT).
- Logistic Regression (LR).

In addition to these ML algorithms, four feed forward neural network (FFNN) architectures were also employed by the IA to do link prediction.

- A $p \times 2p \times 1$ architecture where each layer uses the SELU activation function and p is the number of attributes as shown in equation (1) (NN1).
- A $p \times 5p \times 1$ architecture where each layer uses the SELU activation function (NN2).
- A $p \times 2p \times 2p \times 1$ where the input layer uses RELU and the next ones SELU (NN3).
- A $p \times 5p \times 5p \times 1$ where the input layer uses RELU and the next ones SELU (NN4).

The scaled exponential linear unit (SELU) activation function[3] has a flexible smooth form and can be parameterized by a scaling factor β_0 as shown in (12).

$$\psi(u;\beta_0) \triangleq \begin{cases} \beta_0 u, & u \geq 0 \\ \beta_0 \alpha_0 (e^u - 1), & u < 0 \end{cases} \qquad (12)$$

The rectified linear unit (ReLU) activation function[4] despite its seemingly simple form is very popular because of its adaptability and its easy interpretation. Moreover, the lack of parameters significantly reduces training complexity.

$$\varphi(u) \triangleq \begin{cases} u, & u \geq 0 \\ 0, & u < 0 \end{cases} \qquad (13)$$

It should be noted that both SELU and RELU can be easily found in many ML platforms.

---

[3] www.tensorflow.org/api_docs/python/tf/keras/activations/selu

[4] https://keras.io/api/layers/activations

4 RESULTS

Basic structural properties of the dataset obtained by the Neo4j developer site are shown in table 3. In that table positive examples refer to vertex pairs where a latent edge exists, where negative ones refer to pairs which are not actually linked. Observe that originally there was a considerable imbalance between positive and negative training examples and therefore random downsampling took place.

Table 3: Dataset synopsis.

Property	Value
Negative examples for training data	1580567
Positive examples for training data	105139
Negative examples downsampling	50085
Positive examples downsampling	50085

The benchmark dataset contains authors of scientific papers and their publications over a range of sixty years. In this scenario the objective of link prediction is to discover and recommend potential co-authors in order to boost scientific collaboration. In figure 2 the number of papers entries is shown.

Communication between the Neo4j and the application took place over py2neo, which is one of the most popular Neo4j drivers for Python. It allows sending dynamically formulated Cypher queries and the reception of the results.

The results for each of the basic learning strategies of RF, DT, and LR is shown in table 4. Therein MAE and MSE denote mean absolute error and mean squared error respectively. MSE is especially important as it approximates the performance metric of (5).

Table 4: Performance metrics for the learning strategies.

Metric	RF	DT	LR
Accuracy	0.963	0.963	0.936
Precision	0.965	0.962	0.947
Recall	0.960	0.964	0.924
MAE	0.037	0.037	0.064
MSE	0.037	0.037	0.064
F1	0.963	0.963	0.935

Observe that among these algorithms RF outperforms, even by a small margin, the other two. This can be partly at least attributed to its robustness to outliers and to its averaging nature which typically ensures a more than adequate performance in ML settings.

The receiver operating characteristic (ROC) of an ML model is defined as a way to obtain the probability that the ranking $r(\cdot)$ of any pair of patterns y and y' where $y < y'$ is preserved under that model, namely
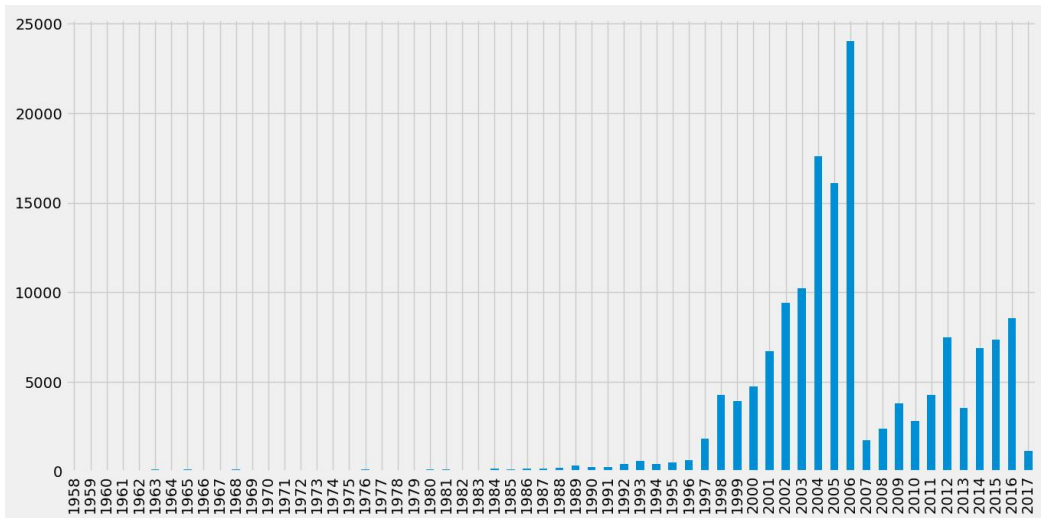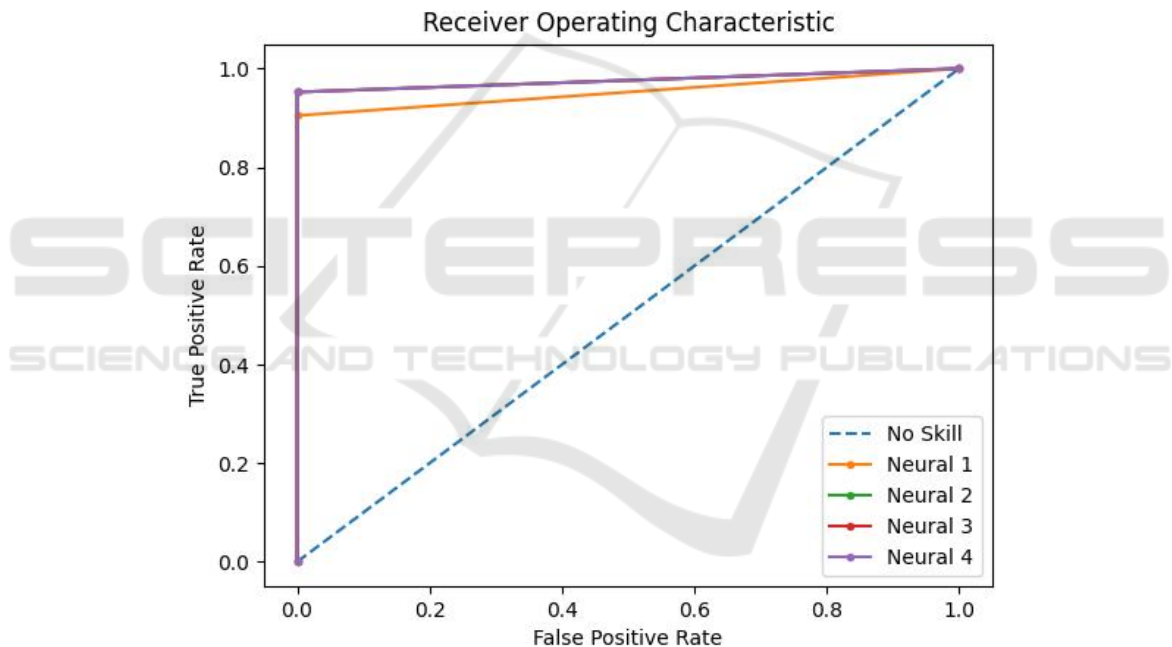
Figure 2: Entries per year.

Figure 3: ROC curves for the FFNN architectures.

$r(y) < r(y')$. Specifically, the area D under the ROC curve yields this probability. This yields (14).

$$D \stackrel{\triangle}{=} \text{prob} \left\{ r(y) < r(y') \mid y < y' \right\} \qquad (14)$$

In figure 3 the ROC curves for the four FFNN architectures is shown. With the exception of NN1, the remaining three architectures achieve comparable performance. This can be explained by the fact that NN2, NN3, and NN4 have more processing neurons than NN1, allowing thus the discovery of more graph structural patterns, especially those of higher order as

graphs are inherently distributed by nature and information is typically not found in a single vertex but instead it is locally spread.

5 CONCLUSIONS

This conference paper focuses on inteligent agent (IA) design utilizing machine learning (ML) techniques for link prediction. The benchmark dataset is a graph containing collaborations between scientists.

Therefore, in this setting link prediction can discover and recommend potential co-authors, enhancing thus research. The proposed IA uses an array of ML strategies including random forest, decision tree, logistic regression, and neural networks. These were trained with graph structural attributes describing the potential of a pair of vertices to attract new neighbors as the higher this potential is, the more probable the vertex pair under consideration is to be connected. These attributes include preferential attachment, Adamic-Adar index, and resource allocation metric.

This work can be extended in a number of ways. First and foremost, the IA can be tested in larger graphs which have a greater variety of structural patterns. Moreover, more ML algorithms can be applied to the same local attributes. Alternatively, the entire graph can be used in ML algorithms natively supporting two-dimensional data such as matrices and images in order to address the link prediction problelm using local and global patterns.

ACKNOWLEDGEMENTS

REFERENCES

Bedre, J. S. and Prasanna, P. (2022). A novel facial emotion recognition scheme based on graph mining. In *ICSCN*, pages 843–853. Springer.

Chen, M., Li, P., Wang, R., Xiang, Y., Huang, Z., Yu, Q., He, M., Liu, J., Wang, J., Su, M., et al. (2022). Multifunctional fiber-enabled intelligent health agents. *Advanced Materials*, 34(52).

Chiatti, A., Bardaro, G., Motta, E., and Daga, E. (2022). A spatial reasoning framework for commonsense reasoning in visually intelligent agents. In *AIC*. CEUR.

Dai, G., Zhu, Z., Fu, T., Wei, C., Wang, B., Li, X., Xie, Y., Yang, H., and Wang, Y. (2022). Dimmining: Pruning-efficient and parallel graph mining on near-memory-computing. In *Annual International Symposium on Computer Architecture*, pages 130–145.

Dong, Y., Ma, J., Wang, S., Chen, C., and Li, J. (2023). Fairness in graph mining: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Drakopoulos, G., Giannoukou, I., Mylonas, P., and Sioutas, S. (2020a). A graph neural network for assessing the affective coherence of Twitter graphs. In *IEEE Big Data*, pages 3618–3627. IEEE.

Drakopoulos, G., Giannoukou, I., Sioutas, S., and Mylonas, P. (2022). Self organizing maps for cultural content delivery. *NCAA*.

Drakopoulos, G., Giotopoulos, K. C., Giannoukou, I., and Sioutas, S. (2020b). Unsupervised discovery of semantically aware communities with tensor Kruskal decomposition: A case study in Twitter. In *SMAP*. IEEE.

Drakopoulos, G. and Kafeza, E. (2020). One dimensional cross-correlation methods for deterministic and stochastic graph signals with a Twitter application in Julia. In *SEEDA-CECNSM*. IEEE.

Drakopoulos, G., Kafeza, E., Mylonas, P., and Al Katheeri, H. (2020c). Building trusted startup teams from LinkedIn attributes: A higher order probabilistic analysis. In *ICTAI*, pages 867–874. IEEE.

Drakopoulos, G., Kafeza, E., Mylonas, P., and Iliadis, L. (2021a). Transform-based graph topology similarity metrics. *NCAA*, 33(23):16363–16375.

Drakopoulos, G., Kafeza, E., Mylonas, P., and Sioutas, S. (2021b). A graph neural network for fuzzy Twitter graphs. In Cong, G. and Ramanath, M., editors, *CIKM companion volume*, volume 3052. CEUR-WS.org.

Drakopoulos, G., Kafeza, E., Mylonas, P., and Sioutas, S. (2021c). Approximate high dimensional graph mining with matrix polar factorization: A Twitter application. In *IEEE Big Data*, pages 4441–4449. IEEE.

Drakopoulos, G., Kanavos, A., Tsolis, D., Mylonas, P., and Sioutas, S. (2017). Towards a framework for tensor ontologies over Neo4j: Representations and operations. In *IISA*. IEEE.

Drakopoulos, G. and Mylonas, P. (2022). A genetic algorithm for Boolean semiring matrix factorization with applications to graph mining. In *Big Data*. IEEE.

Elshan, E., Zierau, N., Engel, C., Janson, A., and Leimeister, J. M. (2022). Understanding the design elements affecting user acceptance of intelligent agents: Past, present and future. *Information Systems Frontiers*, pages 1–32.

Gao, Z., Gama, F., and Ribeiro, A. (2022). Wide and deep graph neural network with distributed online learning. *IEEE Transactions on Signal Processing*, 70:3862–3877.

Giraldo, J. H., Mahmood, A., Garcia-Garcia, B., Thanou, D., and Bouwmans, T. (2022). Reconstruction of time-varying graph signals via Sobolev smoothness. *IEEE Transactions on Signal and Information Processing over Networks*, 8:201–214.

Kaswan, K. S., Dhatterwal, J. S., and Balyan, A. (2022). Intelligent agents based integration of machine learning and case base reasoning system. In *ICACITE*, pages 1477–1481. IEEE.

Li, P., Shlezinger, N., Zhang, H., Wang, B., and Eldar, Y. C. (2022a). Graph signal compression by joint quantization and sampling. *IEEE Transactions on Signal Processing*.

Li, T., Zhou, Z., Li, S., Sun, C., Yan, R., and Chen, X. (2022b). The emerging graph neural networks for intelligent fault diagnostics and prognostics: A guideline and a benchmark study. *Mechanical Systems and Signal Processing*, 168.

Lin, B., Cassee, N., Serebrenik, A., Bavota, G., Novielli, N., and Lanza, M. (2022). Opinion mining for software development: A systematic literature review. *TOSEM*, 31(3):1–41.

Liu, J., Li, D., Gu, H., Lu, T., Zhang, P., Shang, L., and Gu, N. (2023). Personalized graph signal processing for collaborative filtering. In *Web Conference*, pages 1264–1272. ACM.

Moussawi, S., Koufaris, M., and Benbunan-Fich, R. (2023). The role of user perceptions of intelligence, anthropomorphism, and self-extension on continuance of use of personal intelligent agents. *European Journal of Information Systems*, 32(3):601–622.

Narumi, T. et al. (2022). Effect of attractive appearance of intelligent agents on acceptance of uncertain information. In *International Conference on Human-Computer Interaction*, pages 146–161. Springer.

Nefla, O., Brigui, I., Ozturk, M., and Viappiani, P. (2022). Intelligent agents for multi-user preference elicitation. In *Advances in Deep Learning, Artificial Intelligence and Robotics*, pages 151–161. Springer.

Ortega, A. (2022). *Introduction to graph signal processing*. Cambridge University Press.

Pirsa, S., Sani, I. K., and Mirtalebi, S. S. (2022). Nano-biocomposite based color sensors: Investigation of structure, function, and applications in intelligent food packaging. *Food Packaging and Shelf Life*, 31.

Riedl, M. O. (2019). Human-centered artificial intelligence and machine learning. *Human behavior and emerging technologies*, 1(1):33–36.

Ruiz, L., Chamon, L. F., and Ribeiro, A. (2023). Transferability properties of graph neural networks. *IEEE Transactions on Signal Processing*.

Shang, P., Liu, X., Yu, C., Yan, G., Xiang, Q., and Mi, X. (2022). A new ensemble deep graph reinforcement learning network for spatio-temporal traffic volume forecasting in a freeway network. *Digital Signal Processing*, 123.

Shi, J. and Moura, J. M. (2022). Graph signal processing: Dualizing GSP sampling in the vertex and spectral domains. *IEEE Transactions on Signal Processing*.

Singh, M. (2022). Using natural language processing and graph mining to explore inter-related requirements in software artefacts. *ACM SIGSOFT Software Engineering Notes*, 44(1):37–42.

Yoon, M., Gervet, T., Hooi, B., and Faloutsos, C. (2022). Autonomous graph mining algorithm search with best performance trade-off. *Knowledge and Information Systems*, pages 1–32.

Zhao, X., Yao, J., Deng, W., Ding, P., Zhuang, J., and Liu, Z. (2022). Multiscale deep graph convolutional networks for intelligent fault diagnosis of rotor-bearing system under fluctuating working conditions. *IEEE Transactions on Industrial Informatics*, 19(1):166–176.