Applications of Dynamic Heterogeneous Information Networks for AIOps

Rakan Zeid Al Masri, Karlsruhe Institute of Technology

Abstract Real-world systems, such as cloud systems, are often complex, with many different actors interacting with one another in diverse ways, hence difficult to model and expensive to maintain. Recently, many researchers are beginning to model interconnected, multi-typed systems as heterogeneous information networks (HINs) and develop algorithms to extract the deep semantics lying within. A relatively recent development, AIOps is a proposed solution to the maintenance problem. AIOps leverages big data, analytics, and machine learning to automate maintenance and upkeep tasks, thus improving efficiency and lowering costs. This paper provides an overview of heterogeneous information networks and AIOps. I will introduce both topics, along with some real-world examples, and provide my vision of how these seemingly unrelated fields can be linked together to get new insight on network analysis and upkeep.

1 Introduction

Much of today's software infrastructure is hosted on cloud platforms, due to many benefits such as flexibility and reduced operational cost [1]; however, reliance on such systems does come with some drawbacks. Due to their size and complexity, it is often difficult to maintain and analyze such systems. Researchers typically use graphs or networks to model complicated, interconnected systems. We typically refer to such networks, without the loss of generality, as information networks [2].

Contemporary network analysis often overlooks the heterogeneity of real-world systems, thus ignoring the rich semantic information that can be extracted from networks [3]. We call multi-typed, interconnected networks: heterogeneous information networks (HINs for short). As opposed to standard network models (also called, homogeneous information networks), HINs effectively fuse more information and contain rich semantics in nodes and links.

A problem that remains is the maintenance and upkeep of these critical networks. A relatively recent development to solve this problem is AIOps. AIOps uses big data and machine learning to automate maintenance and upkeep tasks, thus improving business operations and saving costs [4].

My aim with this paper is to shine a light on HINs and AIOps, their key concepts, and some real-world examples in order to motivate more research to be done on how these seemingly unrelated fields can be linked together to improve our understanding of networks. The rest of the paper is organized as follows. In Section 2, I will introduce HINs, some necessary definitions, and a classification of the use cases of HINs. Section 3 is about two real-world applications of HIN analysis. Section 4 motivates AIOps and gives an overview of its use cases. In Section 5, I introduce a production system that utilizes AIOps. In Section 6, I motivate linking these two fields in interesting ways. Section 7 concludes this paper.

2 Heterogeneous Information Networks

Heterogeneous information networks are powerful tools that allow us to abstract complex systems and analyze them.

2.1 Basic Concepts and Definitions

In this section, I will introduce some basic but crucial concepts in HIN analysis along with some examples to develop an intuition for HINs.

As previously discussed, information networks are high-level abstractions of real-world networks. They are defined formally as follows:

Definition 1. ***Information Network*** *[2],[3]. An information network is defined as a directed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *with an object type mapping function* $\varphi : \mathcal{V} \rightarrow \mathcal{A}$ *and a link type mapping function* $\psi : \mathcal{E} \rightarrow \mathcal{R}$*. Each object*

$v \in V$ belongs to one particular object type in the object type set $\mathcal{A}$: $\varphi(v) \in \mathcal{A}$ and each link $e \in \mathcal{E}$ belongs to one particular relation type in the relation type set $\mathcal{R}$: $\psi(e) \in \mathcal{R}$. If two links belong to the same relation type, the two links share the same starting object type as well as the ending object type.

Different from traditional network model definitions, we distinguish object types and relationship types in information networks. We differentiate between heterogeneous and homogeneous information networks based on the number of link/object types.

Definition 2. ***Heterogeneous vs. Homogeneous Information Networks***. *The information network is heterogeneous if the types of objects $|\mathcal{A}| > 1$ or the types of edges $|\mathcal{R}| > 1$; otherwise we refer to it as homogeneous.*

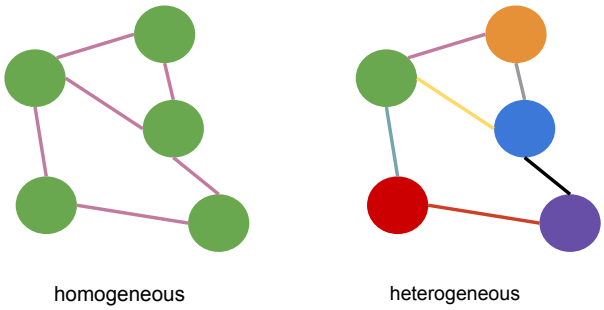

Fig. 1. Homogeneous vs. Heterogeneous networks

Example 1. Fig. 1 illustrates the difference between homogeneous and heterogeneous networks. In the homogeneous network, all the objects are of the same type (i.e., all of them are green) and all the links are of the same type (i.e., all of them are purple); however, in the heterogeneous network, there are many types of objects and edges.

Because heterogeneous information networks are often very complex and large, it is necessary to provide a meta-level (i.e., schema-level) description of the network. That's where the concept of a network schema comes in - it describes the structure (i.e., the types of objects and how they are connected) of the network. Formally, it is defined as follows:

Definition 3. ***Network schema*** *[3]. The network schema, denoted as $S = (\mathcal{A}, \mathcal{R})$, is a meta template for an information network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the object type mapping function $\varphi : \mathcal{V} \rightarrow \mathcal{A}$ and the link type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, which is a directed graph defined over object types $\mathcal{A}$, with edges as relations from $\mathcal{R}$.*

An information network based on a network schema is called a **network instance**.

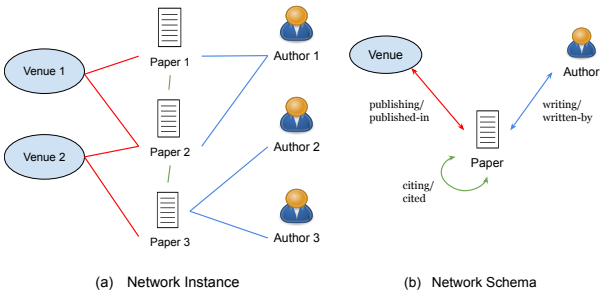

Fig. 2. A network instance (a) and its network schema (b)

Example 2. Fig. 2(a) is an example of a heterogeneous bibliographic network and is based on the network schema, Fig. 2(b). The network schema gives a meta-level description of the structure of the network. In this example, there are three objects: Venue, Author, and Paper. The objects are connected in various ways (for example, a paper is written by an author).

A very compelling reason as to why more researchers are using HINs in data mining is the rich semantic information that can extracted from them. As opposed to standard network models (i.e., homogeneous ones), objects can be connected in different ways and each way or, better yet, path has a different physical meaning. We call these paths: **meta-paths**.

Definition 4. ***Meta path*** *[3]. A meta path $\mathcal{P}$ is a path defined on a schema $S = (\mathcal{A}, \mathcal{R})$ and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$, which defines a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$, where $\circ$ denotes the composition operator on relations.*

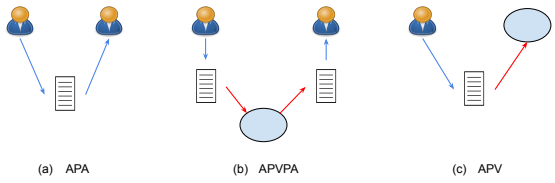

Fig. 3. Examples of meta-paths based on the previous bibliographic network

We typically refer to meta-paths using their object types, unless there are multiple edge types between said objects. For example, in Fig. 3(a), we refer to the composite relation of two authors writing a paper (i.e., $Author \xrightarrow{writing} Paper \xrightarrow{written-by} Author$) as APA. We call this meta-path a **symmetric path**, because the relation is equal to its inverse (i.e., a meta-path $\mathcal{P}$ is symmetric, iff $\mathcal{P} = \mathcal{P}^{-1}$). A concrete

Table 1: Meta-paths, path instances, and their physical meaning based on Fig. 2.

Meta path	Path instance	Physical meaning
APA	$Author_2 - Paper_3 - Author_3$	Authors co-write a paper.
$APVPA$	$Author_1 - Paper_1 - Venue_1 - Paper_2 - Author_1$	Authors publish papers in the same venue.
APV	$Author_3 - Paper_3 - Venue_2$	Author writes a paper and publishes it in a venue.

path $p = (a_1 a_2 \cdots a_l)$ between objects a_1 and a_l in network $\mathcal{G}$ is called a **path instance** of the meta-path $\mathcal{P}$. Two meta-paths $\mathcal{P}_1 = (A_1 A_2 \cdots A_l)$ and $\mathcal{P}_2 = (B_1 B_2 \cdots B_k)$ are **concatenable** iff A_l is equal to B_1, and the concatenated path is written as $\mathcal{P} = (\mathcal{P}_1 \mathcal{P}_2)$, which equals to $(A_1 A_2 \cdots A_l B_2 \cdots B_k)$.

Example 3. As shown in Table 1, there are multiple ways that authors can be connected, and depending on the path taken the meaning changes. This is a key difference between homogeneous and heterogeneous networks. Note that the shorthand notation for meta paths (e.g., APA) is used because there is no ambiguity in the notations (e.g., there is only one type of relation between Author and Paper, vice versa).

Meta-paths are a very crucial part of HIN analysis, because they allow us to extract rich semantics from the network structure and feed them to machine learning algorithms to do various tasks. A description of some of the tasks that HINs are used for comes in the next section.

2.2 Use Cases

HIN analysis can be used in a variety of data mining tasks. The follow is a classification of the various use cases of HINs based on the work from the paper: *A Survey of Heterogeneous Information Network Analysis* [3]. The paper divides the use cases into seven distinct types:

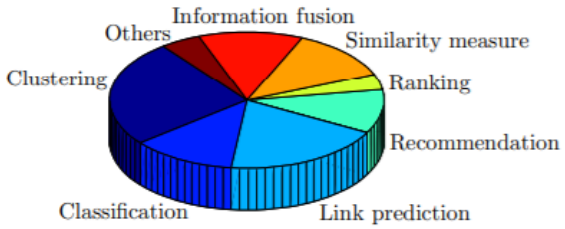

Fig. 4. Paper distribution of heterogeneous information network analysis on different data mining tasks [3]

A. *Similarity Measure*

Similarity measure is to evaluate the similarity of objects based on various characteristics. It is primarily used in tasks such as web search, clustering, and product recommendation.

B. *Clustering*

Clustering is the process of partitioning a set of data objects into a set of clusters, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters.

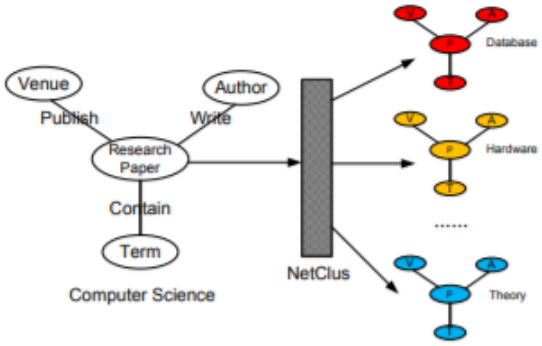

Fig. 5. Clustering on a bibliographic network [5]

C. *Classification*

Classification is a data analysis task where a model or classifier is constructed to predict class (categorical) labels.

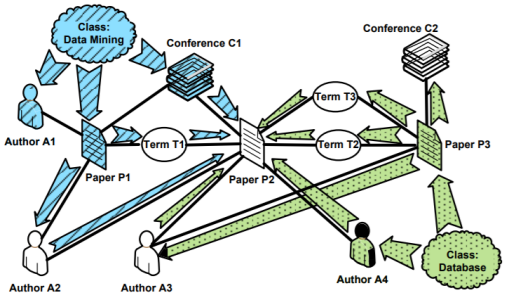

Fig. 6. Classification on a bibliographic network [6]

D. *Link Prediction*

Link prediction is the problem of estimating the likelihood of a link existing between two objects based

on their characteristics.

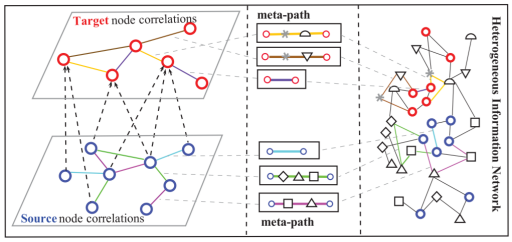

Fig. 7. Collective link prediction in HINs [7]

E. *Ranking*

Ranking is the data mining task, which evaluates object importance or popularity based on some ranking functions.

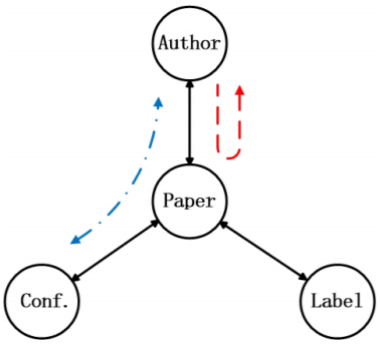

Fig. 8. An example of Ranking [8]

F. *Recommendation*

Recommendation systems help consumers make product recommendation that are likely to be of interest to the user such as books, movies, and restaurants.

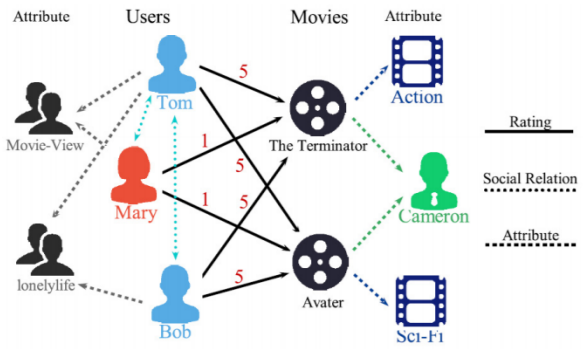

Fig. 9. A movie recommendation HIN [9]

G. *Information Fusion*

Information fusion is the process of merging information from heterogeneous sources with different conceptual, contextual and typographical representations.

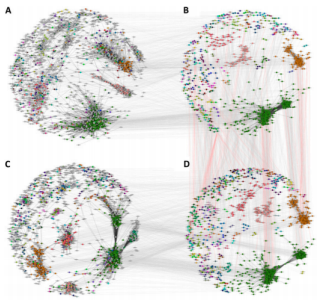

Fig. 10. An example of network alignment [10]

3 Applications of HINs

In this section, I will introduce two interesting real-world applications of HINs. The first paper is about using HINs in Cybersecurity, namely malicious domain detection. The second paper proposes a novel approach of dynamic network embedding that provides a new perspective for heterogeneous data analysis and is practical for real-world applications.

3.1 HinDom

Domain name system (DNS) is a crucial part of the Internet's infrastructure. In short, a DNS translates web domains (e.g., google.com) to IP-addresses (e.g., 168.212.226.204). It is clear to see that such an important system must be kept secure against malicious actors.

Just as the ways that we defend our computers have become more advanced, so too have the techniques that maliciours actors use to bypass modern security systems. In the paper, *HinDom: A Robust Malicious Domain Detection System based on Heterogeneous Information Network with Transductive Classification* [11], the authors propose a new method of malicious domain detection using HINs.

HinDom is based on two assumptions: (i) malicious domains have strong associations between one another; (ii) it is difficult for hackers to distort associations between domains on a large scale (too expensive and/or time-consuming). As shown in Fig. 11, HinDom has five main components: Data Collector, HIN Constructor, Graph Pruner, Meta-path Selector, and Transductive Classifier.

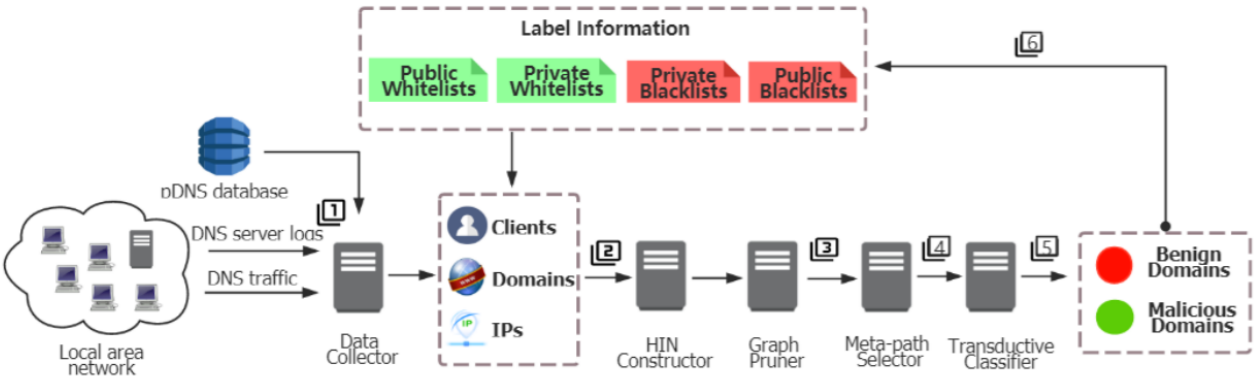

Fig. 11. The architecture of HinDom [11]

A. *Data Collector*

HinDom starts by collecting DNS data during a user-configured time window from various sources, namely: *(i) DNS server logs; (ii) DNS traffic; (iii) Passive DNS database (pDNS for short).* When queried, DNS servers generate various logs that include useful data, such as source IP, time, etc. DNS traffic contains comprehensive information, such as DNS record types. Some organizations aggregate captured DNS messages in the form of pDNS data.

B. *HIN Constructor*

Based on the collected data, HinDom constructs a HIN that abstracts the DNS environment consisting of clients, domains, IP addresses, and the relations between them. The relations are defined as follows:

- **Client-query-Domain:** A client queries a domain.
- **Client-segment-Client:** Both clients belong to the same network segment.
- **Domain-resolve-IP:** The domain is resolved to the IP address.
- **Domain-similar-Domain:** The domains share character-level similarity.
- **Domain-cname-Domain:** Both domains are in a CNAME record.
- **IP-domain-IP:** Both IP addresses are mapped to the same domain.

All of these relations are stored in their respective adjacency matrix and have inverses (i.e., Client-query-Domain and Domain-queried-by-Client).

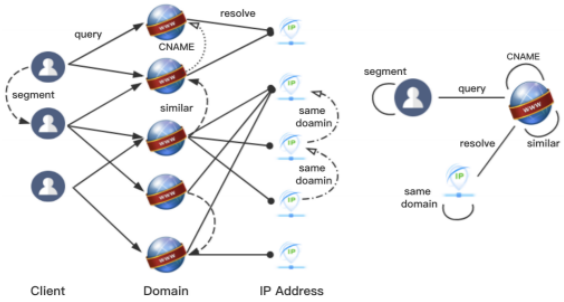

Fig. 12. A network instance of the HinDom network schema [11]

C. *Graph Pruner*

To improve performance, various nodes are filtered out and not modeled. The graph filter/pruner is based on certain rules. For example, popular domains that are queried by large amounts of people are likely to be benign. Some exceptions to the aforementioned rules are set to avoid information loss.

D. *Meta-path Combiner*

Meta-paths are key to be able to fully utilize HINs and denote the complex associations between nodes. HinDom defines meta-paths that begin and end with domains (i.e., symmetric meta-paths with $A_1 = A_{final} = Domain$). They are defined as follows:

- **Domain** $\xrightarrow{similar}$ **Domain:** The researchers noticed that benign and malicious domains differ in character distributions and that malicious domain names from the same family tend to follow a similar textual pattern.

- **Domain** $\xrightarrow{cname}$ **Domain:**
  The CNAME domain of a benign domain is unlikely to be malicious and vice versa.

- **Domain** $\xrightarrow{queried-by}$ **Client** $\xrightarrow{queries}$ **Domain:**
  Clients that are infected by the same attacker tend to query partially overlapping sets of malicious domains.

- **Domain** $\xrightarrow{resolves-to}$ **IP** $\xrightarrow{resolved-from}$ **Domain:**
  Domains resolved to the same IP address tend to belong to the same class.

- **Domain** $\xrightarrow{queried-by}$ **Client** $\xrightarrow{segment}$ **Client** $\xrightarrow{queries}$ **Domain:**
  Neighboring clients are vulnerable to the same attacks.

- **Domain** $\xrightarrow{resolves-to}$ **IP** $\xrightarrow{domain}$ **IP** $\xrightarrow{resolved-from}$ **Domain:**
  Attackers are likely to reuse their domain or IP resources.

The PathSim [12] algorithm, based on the meta-paths defined above, is used to measure the similarity between nodes.

E. *Transductive Classifier*

HinDom applies a meta-path-based transductive classification method to categorize unlabeled domains. Transductive classification utilizes the relatedness of objects to spread labels. Objects are similar when they either share an edge or have "path relations", which, in the context of HINs, are meta-paths [13]. The classification results are then analyzed and added to a private whitelist or blacklist for future detection.

HinDom has been evaluated in two real-world networks: CERNET2 and TUNET, the results of which can be found in the paper. A drawback of HinDom is that it is not a real-time detection system; however, it can be run every hour or day to detect ongoing malicious activity.

3.2 StHNE and DyHNE

In a network embedding problem, the aim is to find low dimensional vector representations of nodes while preserving the network structure. Embedding approaches have several potential advantages, such as increased performance compared to their counterparts [14].

Conventional methods operate on static networks as opposed to dynamic ones, of which HINs are an example of, that change through the addition (or deletion) of nodes. Because tiny changes often influence the entire network, these methods have to reconstruct the entire network, which is very time-consuming.

The authors of the paper *Dynamic Heterogeneous Information Network Embedding with Meta-path based Proximity* [15] propose an efficient network embedding model that is able to handle complex evolutions of HINs effectively while not sacrificing semantics and/or structure.

Before explaining how the architecture works, I must first define a few key terms:

To be able to preserve the semantics and structure of the network, we need to be able to measure the similarity between nodes; that's where meta-path-based first- and second-order proximity come in handy.

First-order proximity indicates the pairwise similarity between nodes, while second-order proximity indicates the similarity between a node and its neighborhood set (i.e., the set of nodes connected to a specific node via some path instance). Formally, they are defined as follows:

Definition 5. ***Meta-path-based First-Order Proximity*** *[15]. For a pair of nodes (v_i, v_j), the number of path instances following the meta-path m connecting them represents the first-order proximity between v_i and v_j.*

Definition 6. ***Meta-path-based Second-Order Proximity*** *[15]. Under a meta-path m, the neighborhoods $\mathcal{N}(v_i)^m$ of node v_i contain nodes connected to v_i via path instances following m. The proximity between v_i's neighborhoods $\mathcal{N}(v_i)^m$ and v_j's neighborhoods $\mathcal{N}(v_j)^m$ is defined as the second-order proximity based on meta-path m.*

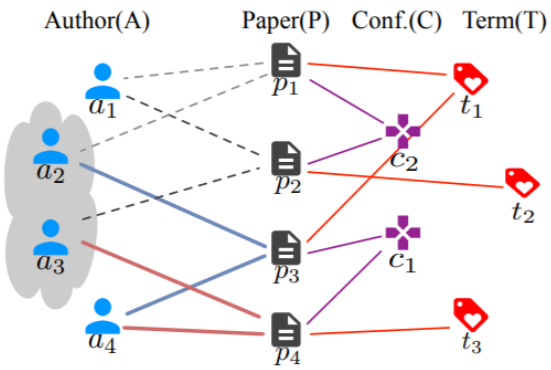

Fig. 13. An example of first- and second-order proximities [15]

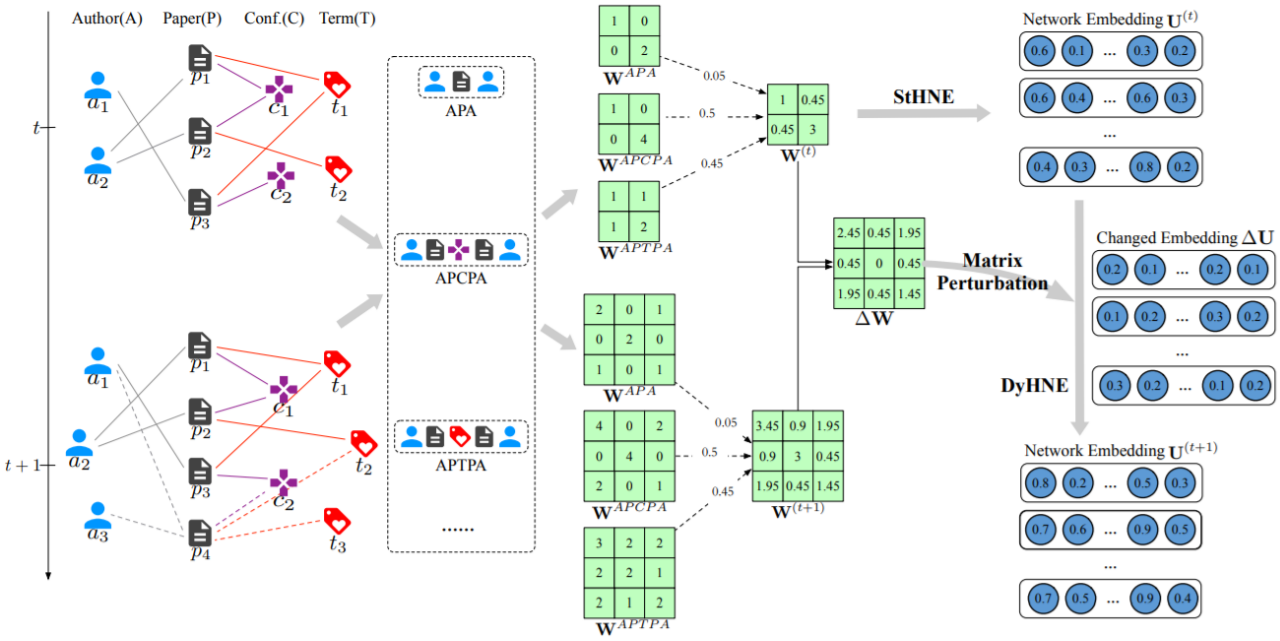

Fig. 14. The StHNE and DyHNE architecture [15]

Example 4. In Fig. 13, meta-path-based first- and second-order proximities are shown. Because nodes a_1 and a_2 are connected via the meta-path instance $a_1p_1a_2$ of the meta-path APA, their vector representations should be placed close together. Under the same meta-path APA, the neighborhood of a_1 is the same as that of a_4 (i.e., a_2, a_3), hence their vector representations should also be close.

To be able to store the proximity information, we define a new matrix that integrates adjacency matrices and meta-paths to capture the structure and semantics of the HIN.

Definition 7. ***Meta-path Augmented Adjacency Matrix*** *[15]. Given a meta-path m, we define a meta-path augmented adjacency matrix as $W^m = [w^m_{ij}]$, where w^m_{ij} is the number of path instances following m connecting nodes v_i and v_j. Naturally, W^m combines the topological structure and semantics of the HIN. And it is symmetric, if m is a symmetric meta-path.*

The architecture is made up of two parts: a static (StHNE) and dynamic (DyHNE) one.

A. *StHNE*

i. Meta-path augmented adjacency matrices based on a defined set of meta-paths are extracted and assigned weights.

ii. The first- and second-order proximities are preserved using equations that represent nodes as d-dimension vectors.

iii. A unified model that combines multiple meta-paths is created that preserves both the first- and second-order proximities. The StHNE problem can be formulated as a minimization problem.

iv. The equation representations of first- and second-order proximities are transformed into generalized eigenvalue problems via spectral theory.

v. The meta-path adjacency matrices are then fused.

vi. Finally, the fused meta-path adjacency matrix, as well as the generalized eigenvalue problem forms of the first- and second-order proximities, are used to transform the StHNE problem to a generalized eigenvalue problem.

B. *DyHNE*

Now that we have a network embedding at time t, matrix perturbation [16, Chapter 7.2] is used to approximate the change of eigenvalues and eigenvectors. We are then able to update the embeddings at time $t + 1$.

In Section 5 of the paper, the StHNE/DyHNE model is tested against various state-of-the-art embedding models and the results show that not only does it outperform them, but it is also much more efficient.

4 AIOps

As businesses become more and more reliant on cloud computing platforms, so do the expectations of the platforms' stability and security increase. To satisfy businesses' demands, cloud providers must continuously expand their operations; however, scalability does generally come at the cost of maintainability and security, so how is the cloud meant to be scalable, yet maintainable and secure? The answer could be AIOps, which stands for *Artificial Intelligence for IT operations*.

AIOps empowers software and service engineers to efficiently build and run scalable online platforms, of which cloud computing is a prominent example [17]. AIOps uses AI and ML (machine learning) techniques to improve the monitoring, operation and upkeep of large distributed systems [18]. It can reduce operational costs and increase customer satisfaction.

4.1 Why do we need AIOps?

The nature of the software industry has fundamentally changed in the past few years. Instead of delivering boxed products, many software companies now provide online services. Software service providers are not only expected to grow their userbase, but also maintain the stability and security of their service whilst continuously improving their offering to compete in the free market [17]. To fulfill these expectations, intelligent methods of operating and maintaining these services are needed.

Gartner, a global research firm, created the term AIOps to address this particular issue [19]. They project that by 2022, 40% of global enterprises will have strategically implemented AIOps solutions to support their IT operations [19].

4.2 Use Cases

Dr. Jorge Cardoso [18] separates AIOps' troubleshooting tasks in distinct categories: (i) anomaly detection; (ii) fault diagnosis; (iii) fault prediction; (iv) fault recovery. These categories go hand-in-hand when it comes to maintaining systems, as shown in Fig. 15.

A. *Anomaly Detection*

Anomaly detection involves defining what normal behaviour is meant to look like and then being able to discern when a system deviates from said behaviour.

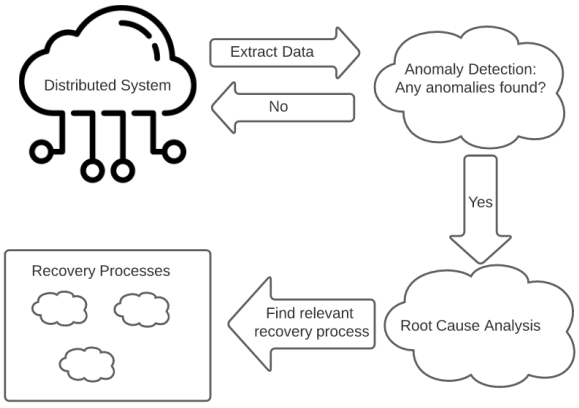

Fig. 15. A conceptual diagram describing how AIOps troubleshooting works

B. *Fault Diagnosis*

After detecting an anomaly, the system needs to find the cause to be able to choose the right recovery process. Fault diagnosis (a.k.a. Root cause analysis) identifies links of dependency that represent causal relationships to the source of an anomaly.

C. *Fault Prediction*

A crucial aspect of intelligent system upkeep is predicting failures before they even happen. Fault prediction is the task of predicting failures using historical data.

D. *Fault Recovery*

Finally, after detecting and diagnosing the anomaly, the system must begin to repair itself. Fault recovery involves exploring decision support systems to manage and select recovery processes to repair failed systems.

5 Predicting Node Failures in the Cloud

Failures in the cloud are hard to detect and estimated to cost $700 billion yearly [20]. To cope with this challenge, AIOps is a proposed solution. For AIOps solutions to be adopted, generally speaking, they must be [1]:

- **Trustable**: Years worth of domain expertise must be incorporated into the Machine learning (ML) models, otherwise engineers cannot trust the solution.
- **Interpretable**: The solutions must be understandable (even at the cost of lower performance).

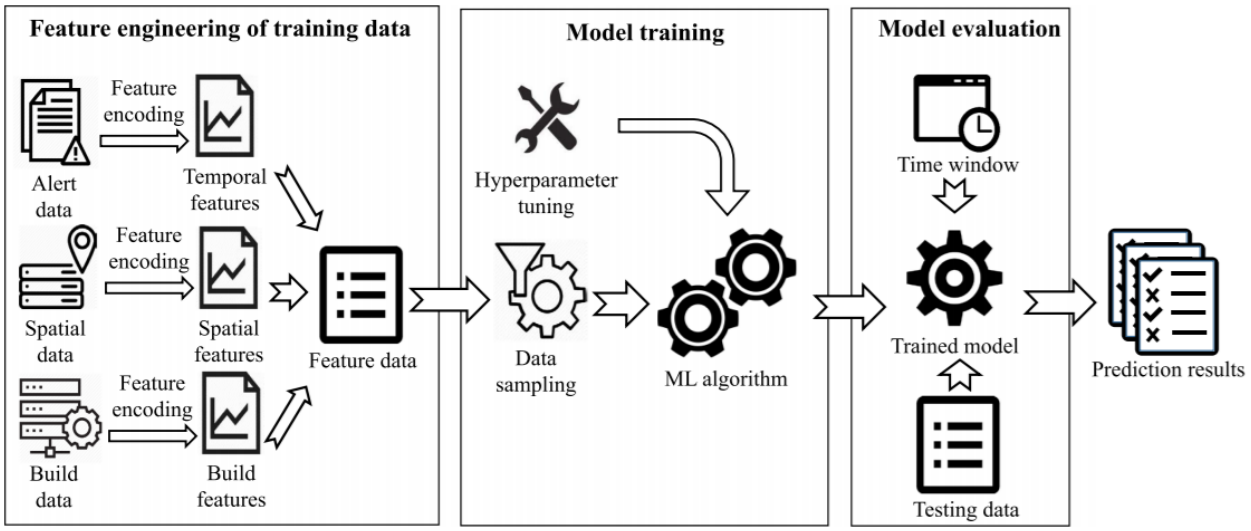

Fig. 16. The AIOps solution pipeline [1]

- **Maintainable**: They must require minimal maintenance and fine-tuning, since one of the main reasons for using AIOps is the lack of human supervision.
- **Scalable**: This is obvious, since AIOps is to be implemented in very large systems.
- **In-context Evaluable**: They must be evaluable in a real-world scenario (traditional ML methods are not realistic).

The authors of *Predicting Node Failures in an Ultra-large-scale Cloud Computing Platform: an AIOps Solution* [1] report their process of building an AIOps solution for predicting node failures for SystemX, a large computing platform at Alibaba, that fulfills all the aforementioned criteria. The researchers describe the challenges of designing the AIOps solution and how they overcame them.

To build the best AIOps solution, the researchers tested various ML algorithm against their data sets and compared the prediction results. Each ML algorithm was put through the pipeline and its results compared with those of others. The pipeline works as follows:

A. *Feature Engineering*

This step focuses on processing monitoring data and producing features to train the ML model. Alert, spatial (i.e., location of the node), and build (i.e., software and hardware configuration of the node) data are processed using feature encoding and combined into feature data.

Dealing with complex data formats of the raw inputs is challenging, so the researchers use different encoding methods depending on the type of data processed.

B. *Model Training*

In this step, the ML model is trained using the feature data produced by the previous step. The researches tested various ML algorithms to determine the best one for their AIOps solution. The models used were: LSTM [21], Random forest [22], and MING (a hybrid approach that combines the latter two) [23]. The algorithms were tested against a baseline model and their relative performance was recorded.

Data skewness was a challenge in this phase, because the chances of nodes failing in a highly reliable system like SystemX are low, hence making it difficult to build accurate ML models. Another challenge in this phase was hyperparameter tuning. Some of the ML algorithms have many parameters that need to be set by the user, so finding the optimal configuration is quite difficult.

C. *Model Evaluation*

Choosing how to evaluate each ML model is crucial in order to find the best one; however, conventional evaluation techniques are not accurate enough. To combat this, the researchers developed their own evaluation technique. The main differences between their technique and conventional ones are:

- **Per-node prediction evaluation**: Instead of evaluating individual prediction results, they evaluate performance of many predictions within a certain time frame. The individual predictions are then considered together to decide if the model was effective.
- **Just-in-time prediction**: Predictions should be

made within a certain time window, because early predictions would lead to financial losses, since nodes that are perfectly functional might be replaced before their lifespan ends, and late predictions would be useless, since DevOps engineers would not have time to react.

After various experiments, the researchers concluded that the random forest algorithm outperformed the other ones and was much more efficient. They claim that their proposed solution has fulfilled the aforementioned criteria in the following ways:

- **Trustable**: The proposed solution outperforms other ones and leverages domain expertise, because the parameters in the ML models are easily modifiable.
- **Interpretable**: The chosen ML model (random forest) is not black-box and hence interpretable.
- **Maintainable**: The pipeline stages can all be automated, but a challenge that remains is hyperparameter tuning.
- **Scalable**: The solution has already been adopted in Alibaba's large scale system, SystemX, hence it is scalable.
- **In-context Evaluable**: Proposed evaluation techniques allow for an accurate, context-aware evaluation of the system.

6 Linking HINs to AIOps

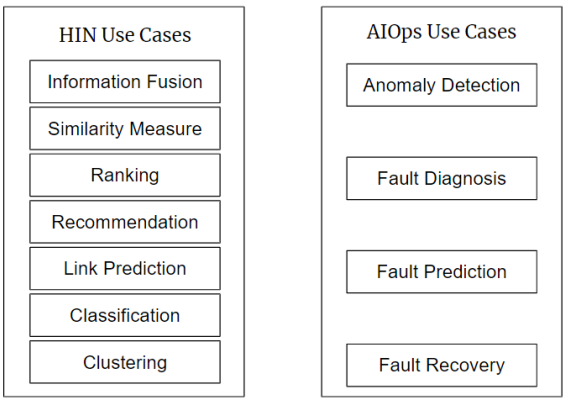

Fig. 17. HIN and AIOps use cases

I have shown how useful and powerful HINs and AIOps are in their own right; however, I believe that, if used together, they can enable users/companies to solve interesting new problems and further develop existing solutions.

Fig. 17 illustrates the HIN and AIOps use cases covered in Sections 2.2 and 4.2, respectively. At first look, it is clear that some HIN and AIOps use cases, namely *Link prediction* and *Fault prediction*, share some similarity. In essence, they solve the same problem: we are attempting to predict the likelihood of an event (a link existing, in the case of *link prediction*, and an error occurring, in the case of *fault prediction*) given some background information or a model.

The AIOps system from Section 5 is a good example of a solution that could be further developed using HINs. If we are able to model (i.e., create a network schema) the node scene, we could analyze the system using HIN techniques and extract some useful information (for example, the root cause of the errors). Modelling the node scene shouldn't be too difficult, since existing HIN systems, like HinDom, are quite similar to it.

There are, however, some challenges to this approach. Although there are quite a few publications about HINs and AIOps separately, there are none that I could find that talk about both fields together. Because I do not perform any experiments in this paper, but rather summarize findings from other papers, I can not unequivocally claim that linking HINs to AIOps is beneficial without any drawbacks. For example, the overhead of HINs, though it can be improved via network embedding, may prove to be too costly for real-time AIOps systems, thus nullifying any benefits gained. I still believe that there is some potential and hope that this paper motivates some research to be done that links these two fields.

7 Conclusion

In this paper, I introduced AIOps and HINs, explained their basic concepts and terminology, and gave an overview of their use cases. In Section 3, I introduced two applications of HINs, namely *HinDom* and *StHNE/DyHNE*. HinDom is an intelligent malicious domain detection system that uses HINs to model the DNS scene, while StHNE/DyHNE is a network embedding model that is more efficient than conventional models. I later introduced a fault prediction AIOps system that is scalable and efficient. Finally, I shared my thoughts on how these two fields can be linked together.

References

[1] Y. Li et. al. "Predicting Node Failures in an Ultra-Large-Scale Cloud Computing Platform: An AIOps Solution". In: 29.2 (2020). ISSN:

1049-331X. DOI: 10.1145/3385187. URL: https://doi.org/10.1145/3385187.

[2] Y. Sun and J. Han. "Mining Heterogeneous Information Networks: A Structural Analysis Approach". In: *SIGKDD Explor. Newsl.* 14.2 (Apr. 2013), pp. 20–28. ISSN: 1931-0145. DOI: 10.1145/2481244.2481248. URL: https://doi.org/10.1145/2481244.2481248.

[3] C. Shi et. al. "A survey of heterogeneous information network analysis". In: *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2017), pp. 17–37. DOI: 10.1109/TKDE.2016.2598561.

[4] A. Lerner. *AIOps Platforms*. URL: https://blogs.gartner.com/andrew-lerner/2017/08/09/aiops-platforms/. (accessed: 23.02.2021).

[5] Y. Sun, Y. Yu, and J. Han. "Ranking-Based Clustering of Heterogeneous Information Networks with Star Network Schema". In: KDD '09. Paris, France: Association for Computing Machinery, 2009, pp. 797–806. ISBN: 9781605584959. DOI: 10.1145/1557019.1557107. URL: https://doi.org/10.1145/1557019.1557107.

[6] M. Ji et. al. "Graph Regularized Transductive Classification on Heterogeneous Information Networks". In: *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2010, pp. 570–586. ISBN: 978-3-642-15880-3.

[7] B. Cao, X. Kong, and P. S. Yu. "Collective Prediction of Multiple Types of Links in Heterogeneous Information Networks". In: *2014 IEEE International Conference on Data Mining*. 2014, pp. 50–59. DOI: 10.1109/ICDM.2014.25.

[8] Y. Li et. al. "HRank: A Path Based Ranking Method in Heterogeneous Information Network". In: *Web-Age Information Management*. Cham: Springer International Publishing, 2014, pp. 553–565. ISBN: 978-3-319-08010-9.

[9] C. Shi et. al. "Semantic Path Based Personalized Recommendation on Weighted Heterogeneous Information Networks". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15. Melbourne, Australia: Association for Computing Machinery, 2015, pp. 453–462. ISBN: 9781450337946. DOI: 10.1145/2806416.2806528. URL: https://doi.org/10.1145/2806416.2806528.

[10] S. P. Ficklin and F. A. Feltus. "Gene coexpression network alignment and conservation of gene modules between two grass species: maize and rice". In: *Plant Physiology* 156.3 (2011), pp. 1244–1256.

[11] X. Sun, M. Tong, and J. Yang. "Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification". In: *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*. 2019, pp. 399–412.

[12] Y. Sun et. al. "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks". In: *Proceedings of the VLDB Endowment* 4.11 (2011), pp. 992–1003.

[13] X. Li et. al. "On transductive classification in heterogeneous information networks". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 811–820.

[14] W. Nelson et. al. "To Embed or Not: Network Embedding as a Paradigm in Computational Biology". In: *Frontiers in Genetics* 10 (2019), p. 381. ISSN: 1664-8021. DOI: 10.3389/fgene.2019.00381. URL: https://www.frontiersin.org/article/10.3389/fgene.2019.00381.

[15] X. Wang et. al. "Dynamic heterogeneous information network embedding with meta-path based proximity". In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[16] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013. ISBN: 9781421407944. URL: https://books.google.ae/books?id=X5YfsuCWpxMC.

[17] Y. Dang, Q. Lin, and P. Huang. "AIOps: real-world challenges and research innovations". In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE. 2019, pp. 4–5.

[18] J. Cardoso. *Distributed Trace & Log Analysis using ML*. URL: https://www.slideshare.net/JorgeCardoso4. (accessed: 23.02.2021).

[19] P. Prasad and C. Rich. *Market Guide for AIOps Platforms*. URL: https://www.gartner.com/en/documents/3971186/market-guide-for-aiops-platforms. (accessed: 23.02.2021).

[20] M. Machowinski. *Businesses Losing $700 Billion a Year to IT Downtime, Says IHS*. URL: https://technology.ihs.com/572369/businesses-losing-700-billion-a-year-to-it-downtime-says-ihs. (accessed: 12.03.2020).

[21] K. Greff et. al. “LSTM: A search space odyssey”. In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.

[22] L. Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[23] Q. Lin et. al. “Predicting Node Failure in Cloud Service Systems”. In: *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2018. Lake Buena Vista, FL, USA: Association for Computing Machinery, 2018, pp. 480–490. ISBN: 9781450355735. DOI: 10.1145/3236024.3236060. URL: https://doi.org/10.1145/3236024.3236060.