MODELING, SEARCHING, AND EXPLAINING ABNORMAL INSTANCES IN MULTI-RELATIONAL NETWORKS

by

Shou-de Lin

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

September 2006

Table of Contents

List of Tables

List of Figures

Abstract

An important research problem in knowledge discovery and data mining is to identify abnormal instances. Finding anomalies in data has important applications in domains such as fraud detection and homeland security. While there are several existing methods to identify anomalies in numerical datasets, there has been little work aimed at discovering abnormal instances in large and complex relational networks whose nodes are richly connected with many different types of links. To address this problem we designed a novel, unsupervised, domain independent framework that utilizes the information provided by different types of links to identify abnormal nodes. Our approach measures the dependencies between nodes and paths in the network to capture what we call “semantic profiles” of nodes, and then applies a distance-based outlier detection method to find abnormal nodes that are significantly different from their closest neighbors. In a set of experiments on synthetic data about organized crime, our system can almost perfectly identify the hidden crime perpetrators and outperforms several other state-of-the-art methods that have been used to analyze the 9/11 terrorist network by a significant margin.

To facilitate validation, we designed a novel explanation mechanism that can generate meaningful and human-understandable explanations for abnormal nodes discovered by our system. Such explanations not only facilitate the verification and screening out of false positives, but also provide directions for further investigation.

The explanation system uses a classification-based approach to summarize the characteristic features of a node together with a path-to-sentence generator to describe these features in natural language. In an experiment with human subjects we show that the explanation system allows them to identify hidden perpetrators in a complex crime dataset much more accurately and efficiently. We also demonstrate the generality and domain independence of our system by applying it to find abnormal and interesting instances in two representative natural datasets in the movie and bibliography domain. Finally, we discuss our solutions to several related applications including abnormal path discovery, local node discovery, automatic node description and explanation-based outlier detection.

Chapter 1

Introduction

A discovery is said to be an accident meeting a prepared mind.

Albert Szent-Gyorgyi

Discovery has played an important role throughout human history. It is not a one-time, one-place activity, but a process in which scientists have continued to come up with new ideas and theories for thousands of years. As a computer scientist, an interesting question to ask is whether there exists a way for us to model the discovery process, and furthermore, whether it is possible to develop an artificial intelligence (AI) system that can mimic or assist human beings in this process.

To motivate our approach, let us start with the well-known story about the discovery of the theory of Natural Selection. During a five-week trip to the Galápagos Islands, Charles Darwin observed that finches on different islands usually have different beak sizes and shapes. As it is generally known that all finches on the Galápagos Islands are from the same origin, this observation inspired his idea of Natural Selection: finches that are better suited for the environment of a specific island tend to survive better than others on that island. In essence, it was the abnormal characteristic of the birds that triggered his deeper thinking about evolution, and it was the process of finding reasonable explanations for them that eventually grew into a novel and important theory. The cartoons in Figure 1.1 depict this discovery process

in four stages: problem, inspiration, reasoning, and new theory. In the first stage, one develops a certain question in mind (e.g. how evolution works). In the inspiration stage, one encounters interesting information related to the problem (e.g. different types of beaks). In the third reasoning stage, one tries to make sense out of the unusual observations by generating some hypotheses to explain them. Finally, the explanation or hypothesis can be tested and sharpened, and might become a new theory or discovery (e.g. Natural Selection). Such discovery process is not necessarily a linear journey through these steps–sometimes digressions to the earlier stages is necessary before the final goal is reached.

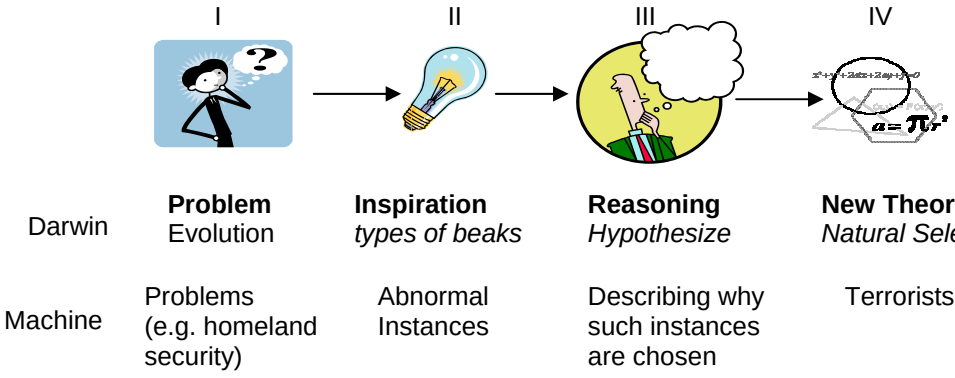


Figure 1.1: An inspiration-driven discovery process

1.1 Problem Definition

The central question we will pursue throughout this thesis is whether and how an AI program can model such a process to perform or assist humans to perform automatic discovery. To be more concrete, we focus on modeling the second and part of the third stage in the above discovery process. We develop a general framework

that can automatically identify abnormal instances in data and explain them, with the goal to point out to humans a set of potentially interesting things in large, complex datasets. Note that there are three key features in this problem. First, our discovery targets data in the form of *multi-relational networks* or *semantic graphs* which allow the representation of complex relationships between objects of different types. Second, we are interested in *abnormal* nodes in these networks instead of central or important ones. Third, we want the discovery system to be able to *explain* its findings in a human-understandable form. We will discuss these aspects in more detail below.

1.2 Multi-relational Networks

The data structure we will focus on is the multi-relational network (MRN). A multi-relational network is a type of network where nodes represent objects of different types (e.g., persons, papers, organizations, etc.) and links represent binary relationships between those objects (e.g. friend, citation, etc.). The term *multi-relational* emphasizes the fact that we focus on networks containing multiple different types of links. A multi-relational network is a powerful representation structure which can encode semantic relationships between different types of objects. It is conceptually similar to other representations such as semantic graphs and semantic networks. For example, a bibliography network such as the one shown in Figure 1.2 is an MRN, where the links represent multiple, different relationships between nodes–fro example, authorship (a link connecting a person and a paper) or citation (a link connecting two paper nodes). Generally, a node in an MRN might also have some attributes associated with it. For example, a *person* node might have *age*

and *weight* as attributes. Though the examples we use throughout this thesis assume there are no such attributes for simplicity reasons, the methodology we will describe can be easily adapted to MRN's that contain node-associated attributes.

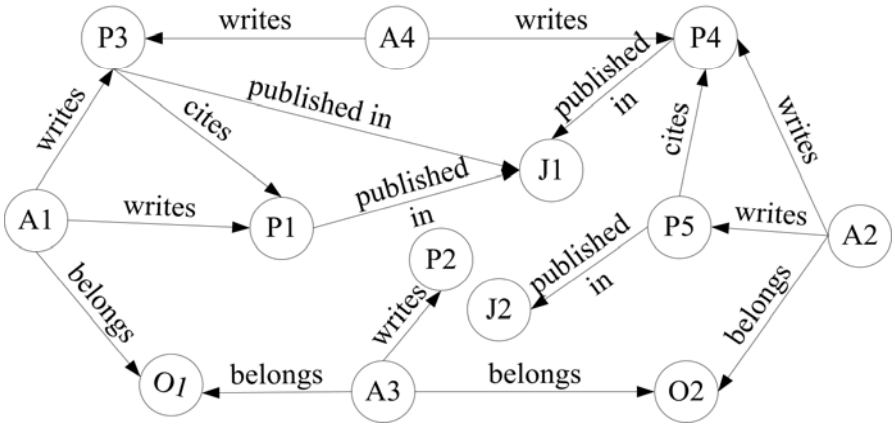


Figure 1.2: A multi-relational bibliography network. The capital letter of each node represents its type: A(author), P(paper), O(organization), J(journal)

MRN's have become a popular method to capture relationship information. For example, a social network or semantic network can be regarded as a multi-relational network in that it has multiple different types of relations. The bibliography network in Figure 1.2 is an MRN. A kinship network is an MRN that represents human beings as nodes and the various kinship relationships between them as links. The Web can be represented as an MRN if we distinguish, for example, incoming, outgoing, and email links. WordNet is an MRN that captures the lexical relationships between concepts. Data stored in relational databases or the ground facts in a knowledge base can often be described via MRN's as well (transformations might be necessary to map n-nary relations onto binary links). Because multi-relational networks are relatively simple but powerful and intuitive way to encode relational information between objects, they

are becoming an important representation schema for analysts in the intelligence and law enforcement communities (M. Sparrow 1991; T. Senator and H. Goldberg 2002; D. Jensen, M. Rattigan et al. 2003) highlight some of the major application domains for our system. Having multiple relationship types in the data is crucial, since different relationship types carry different kinds of semantic information, allowing us to capture deeper meaning of the instances in the network in order to compare and contrast them automatically. Unfortunately, state-of-the-art automated network analysis algorithms such as PageRank (S. Brin and L. Page 1998) or centrality theory (S. Wasserman and K. Faust 1994) cannot deal with relation types in a network, which makes them not as well suited for the type of problems we are trying to solve.

1.3 The Importance of Abnormal Instances

There are a variety of things one can discover from a network. For example, one can try to identify central nodes, recognize frequent subgraphs, or learn interesting network property. Centrality theory (S. Wasserman and K. Faust 1994), frequent subgraph mining (J. Ramon and T. Gaertner 2003) and small world phenomenon (J.M. Kleinberg 2000) are among the well-known algorithms aimed at solving these problems.

The goal of this thesis is different. We do not focus on finding central instances or pattern-level discovery. Instead we try to discover certain *individuals or instances* in the network that look *different* from others. There are three reasons to focus on discovering these types of instances in an MRN. First, we believe that these kinds of instances can potentially play the “light bulb” role depicted in Figure 1.1, in the sense

that something that looks different from others or from its peers has a higher chance to attract people's attention or suggest new hypotheses, and the explanation of them can potentially trigger new theories. The second reason is that there are a number of important applications for a system that can discover abnormal nodes in an MRN, as will be elaborated below. Finally, this is a very challenging problem and so far we are not aware of any system that can utilize the relational information in an MRN to perform anomaly detection.

Application 1: Information Awareness and Homeland Security

Some post-event analyses of the 9/11 attacks show that the implicit and explicit relationships of the terrorists involved do form a relatively large covert network, containing not only the threat individuals but also a large number of innocent people (V. Krebs 2001). Moreover, the persons in the network usually have a variety of connections with each other (e.g. friendships, kinships, business associations, etc). Consequently, it makes sense to represent all this information in a large and complex MRN. This type of data usually contains primarily innocent and benign persons who do not need to perform certain special actions to pursue threat missions. That is to say, it is reasonable to infer that people who look typical (i.e., who have a lot similar peers) are not likely to be malicious. Although threat individuals might try to disguise their actions to avoid detection, it is likely that subtle differences still exist, since they still need to perform unusual actions in order to execute their mission. Such behavior is more likely to create unusual evidence, and, therefore, a node with an abnormal evidence profile has a higher chance to be suspicious compared with ones that have

many similar peers. Our model exploits the deeper information provided by the MRN with the goal to identify the suspicious individuals. Furthermore, since false positives generated by a security and information awareness system (i.e., marking innocent people as suspects) can cause a lot of damage to blameless people, an explanation mechanism which describes why one is chosen becomes very important because it provides investigators with a way to validate the results.

Application 2: Fraud Detection and Law Enforcement

Similar to the previous application, the fraud detection and law enforcement domains also offer huge amount of information about the relationships and transactions among persons, companies or organizations. Being able to identify and explain abnormal instances in such a network could be an important tool for police or investigators to detect criminals and fraudsters. In one of our experiments, we show that our system can successfully identify hidden crime organizers in a complex dataset which other state-of-the-art network algorithms failed to find.

Application 3: General Scientific Discovery

As described in Figure 1.1, abnormal instances and their explanations may also provide some interesting ideas or hints for scientists. In the domain of biology or chemistry, for example, genes, proteins and chemicals can interact with each other in a number of ways, and these interactions can be described by a multi-relational network. One application for our system is to provide new insights to scientists by pointing out unusual biological or chemical connections or individuals. In an ongoing collaboration with biologists, we have created a tuberculosis network where the nodes

are tuberculosis-related genes and the links correspond with how they interact with each other (e.g. activate, depress, etc). The goal is to identify specific genes or proteins whose roles in the network are different from the others, and then provide the system's findings to the domain experts to see if they can lead to further discovery.

Application 4: Data Cleaning

An abnormal record in a fairly accurate database can also be interesting because it might represent an error, missing record, or certain inconsistency. We can apply our system to a relational dataset (which can be represented by a relational graph), to mine for potential errors. For example, in one of our experiments on a movie dataset, the system found that the node "Anjelica Huston" was abnormal. The explanation automatically produced by our system indicated that what makes her abnormal is that she is a parent of some movie person but she never had a child working in the movie industry. The explanation indicated a contradiction, and it turns out that there was a missing record in the "child" relation for this node.

1.4 Explanation

One major concern for a knowledge discovery system rests in the difficulty of verification. Unlike a learning system, a discovery system by definition aims at finding something that was previously unknown. Since the discoveries are previously unknown, it is generally not clear how they can be verified (S. Lin and H. Chalupsky 2004). This concern becomes even more serious for systems applied to security-related problems. As pointed out by (B. Simons and E. H. Spafford 2003), false

positives are a very serious issue for any homeland security system, since even a system with almost perfect 99% accuracy can still result in the mislabeling of millions of innocent individuals when applied to a large population.

Achieving 99% precision without completely sacrificing recall is already an extremely difficult goal for any automated data analysis system, yet it still does not answer the false positive problem. While an unsupervised discovery system like ours has the potential of being able to identify suspicious individuals whose characteristics were previously unknown, it runs an even higher risk of increasing the number of false positives, since there is no pre-existing knowledge to learn from. To deal with this problem we propose the concept of *explanation-based discovery*, where an explanation mechanism is attached to the discovery framework that enables it to produce human-understandable explanations for its findings. With these explanations, users can more intuitively judge the validity of the generated discoveries. An explanation-based discovery system can therefore not only discover things of interest, but also provide a certain amount of information for users to verify the results as well as guide further investigation. This significantly improves such a system's usability and credibility for real-world problems.

1.5 Design Considerations

Until to now we have briefly described the goal and the motivation for our work. In this section we discuss several critical decisions we made while designing the system as well as the requirements we plan to address.

1.5.1 Rule-based and Supervised Systems vs. Unsupervised Systems

There are three major strategies one can apply to identify abnormal instances in MRN's: rule-based learning, supervised learning, and unsupervised learning. Rule-based systems perform some type of pattern-matching based on rules to identify abnormal or suspicious instances. To elaborate on the bibliography MRN example, one could write a rule like "an author node is abnormal if it does not cite any other people's papers". Supervised learning systems take a set of manually labeled abnormal example individuals as inputs, and then try to learn patterns or other descriptions to classify new individuals. One can use known criminal individuals as training examples to learn a classifier for abnormal instances. The advantage of these approaches is that they can achieve relatively high precision due to the manual crafting of rules or selection of training examples. Their main disadvantages are that they are domain dependent, the rules or examples are expensive to create, and, most importantly, that they are very sensitive to human bias. To further elaborate, the only abnormal instances such a system can find are those already in the rule writers or human annotators' mind, which might not be very novel and have less chance to trigger further discovery. We argue that these kinds of abnormal instances are not the ideal candidates to serve as the "light bulb" in stage II of Figure 1.1. Note that Darwin did not *foresee* in advance that it is the variance of beaks that triggers the idea of Natural Selection.

For these reasons, this thesis focuses on finding another type of abnormal instance– is a node that plays different roles in the semantic graph when compared

with others. Because we now define abnormality by comparing and contrasting nodes in the graph, it is possible to design an unsupervised system to identify abnormal nodes. An important advantage of an unsupervised system is that it can be easily adapted to a new domain, with no need to produce a completely new set of rules or training examples (which could be very expensive and time consuming). Another advantage is that such a system is more suitable for security-related problems where the malicious individuals constantly adapt their behavior and methods to avoid capture.

1.5.2 System Requirements

To achieve the goals described above, we believe such an unsupervised, discovery-driven system should satisfy the following requirements:

1. It must utilize as much information provided in the MRN as possible such as the type information of links.
2. It must not rely on manually crafted rules or training examples, and should preferably be easily adapted to new, different domains.
3. Its discoveries and discovering mechanism must be explainable in a meaningful and human understandable manner.
4. It must be scalable enough to handle networks that contain large numbers of nodes, links and link types.

Besides these requirements, there are also some optional features we would like our system to have:

5. It should allow the expression of high-level bias from users. For example, we might want the system to only look for abnormal nodes that are involved in at least one murder event.
6. It should support different levels of detail in its explanations, trading simplicity with informativeness.

1.6 Research Objective and Challenges

The research goal of this thesis is twofold. The first part, which we refer to as the *discovery* stage, is to design a framework that is capable of identifying abnormal nodes in large and complex networks. The second, which we call the *explanation* stage, focuses on designing an explanation mechanism to produce human understandable descriptions (e.g. English) for the discovered nodes. For example, in one our experiments, the system takes an organized crime network as input and then identifies abnormal individuals that are likely to involve in high-level events of interests. It also produces explanations for them such as "*X is abnormal, because it is the only Mafiya group in the dataset that has a member who ordered a contract murder"*.

The main challenge of this task is to design an anomaly detection system for MRN's that can achieve all the requirements discussed previously. To our knowledge, there is no existing algorithm that can simultaneously satisfy both requirements 1 and 2, not to mention the other requirements such as explanation generation. While there are systems aimed at identifying suspicious instances in MRN's (D. Jensen, M. Rattigan et al. 2003; J. Wang, F. Wang et al. 2006), all of them are either rule-based

or supervised classification systems. Conventional outlier and anomaly detection algorithms are unsupervised, but they can only deal with propositional or numerical data, and not MRN's. State-of-the-art unsupervised network algorithms such as PageRank (S. Brin and L. Page 1998), HITS (J.M. Kleinberg 1999), and random walk (B. D. Hughes 1995) are not suitable because they do not take link types into account. We will show in the experiment that the information provided by link type is very important for both identifying the suspicious nodes also for generating explanations. As a result, our system outperforms those unsupervised network algorithms by a large margin in terms of identifying suspicious hidden crime organizers.

Another challenge lies in the fact that it is necessary to consider explanations for the discovered results. That is to say, while designing the discovery system, one needs to take into account how human understandable explanations for them can be generated. We need a model that is complex enough to utilize all the information provided in the semantic graph but not so overly complicated or opaque that it prevents generating human understandable explanations.

1.7 Approach

To deal with these problems, we first need to design a *model* that can capture the semantics of nodes, based on which the system can then *find* nodes with abnormal semantics and *explain* them. We apply both symbolic and statistical methods to exploit the concept of syntactic semantics (W.J. Rapaport 2002) and exploit the complex structure of the network to model the semantics of nodes in an unsupervised manner. This is done by automatically selecting a set of relevant *path types* in the

network to use as semantic features, and then computing statistical dependency measures between nodes and path types to use as feature values.

Given the features of nodes (which we refer to as *semantic profiles*), we then apply a distance-based outlier detection to identify the abnormal nodes. An explanation of an abnormal node is then generated by applying a classification method to separate if from the other nodes in the network, and then translating the resulting classification procedure (i.e., rules generated by the classifier) into a natural language description.

1.8 Contributions

The major contributions of this thesis are:

1. A novel unsupervised framework to identify abnormal instances in a multi-relational network that satisfies the requirements stated above.
2. A set of evaluations of our discovery system showing that it outperforms state-of-the-art unsupervised network algorithms by a large margin.
3. A novel mechanism to generate meaningful and understandable explanations for abnormal instances in multi-relational networks.
4. A human study to evaluate the explanation system which shows that our system allowed human subjects to perform complex data analysis in a significantly more accurate and efficient manner.
5. Demonstration of the generality and applicability of our system by showing that it can identify abnormal and sometimes interesting individuals on a set of representative natural datasets.

As our design has prioritized generalization and domain independence, the individual technologies we created can typically be applied to other problems. This leads to several additional, supplemental contributions for this thesis. Specifically, we will discuss four related problems or applications along with our solutions to them and some experimental results: identifying abnormal local nodes, finding abnormal paths, feature identification and explanation for arbitrary nodes, and explanation-based outlier ranking.

1.9 Thesis Outline

The remainder of the dissertation is organized as follows: Chapter 2 presents a framework to identify abnormal nodes in an MRN based on a hybrid approach that integrates symbolic and statistical methods. Chapter 3 describes experiments to evaluate the framework on a synthetic organized crime dataset. Chapter 4 describes our explanation mechanism for discovered nodes and a human study evaluating the mechanism. Chapter 5 demonstrates a set of abnormal and interesting results our system finds in two representative natural datasets. Chapter 6 discusses several issues such as complexity, incremental and sampling versions of our algorithms and additional applications. Chapter 7 describes related work and Chapter 8 concludes and proposes future research directions.

Chapter 2

Modeling and Finding Abnormal Nodes

In this chapter we describe a framework for identifying abnormal nodes in a multi-relational network. Our goal is to develop an unsupervised framework that utilizes the information provided by the MRN to identify abnormal instances. The central question we need to ask regarding this is the following: how can we define and measure abnormality in a context where each node *is* different from every other (in the sense that they have different names, occupy different positions and connect to different nodes through different links in the network)?

One plausible answer for this question is: a node is abnormal if there are no or very few nodes in the network similar to it based on some similarity measure. Given this definition, the next question that needs to be answered is: how do we measure the similarity of nodes? The following is a list of possible proposals that node similarity could be based on using the information provided by an MRN:

- The type of the nodes (e.g. two nodes are similar if they are of the same or similar type)
- The type of directly connected nodes (e.g. two nodes are similar if they are connected to similar types of nodes)

- The type of indirectly connected nodes via path of certain length (e.g. two nodes can be similar if the sets of the nodes that are within three steps away are similar)
- The type of their links or edges (e.g. two nodes are similar if they have the same types of links)
- The type of indirectly connected links via path of certain length (e.g. two nodes can be similar if the sets of links that are within three steps away are similar)
- The network statistics of the nodes (e.g. two nodes are similar if they have the same degree, or connect to the same amount of nodes)

However, this above proposal seems to be capture only partial connectivity information about the nodes and might not be able to capture the deeper meaning of the nodes. Our real interest is to find the nodes that contain abnormal *semantics* or play different *roles* in the network. Given that our goal is to build a domain-independent and unsupervised system, we believe the best way for us to model the semantics of the nodes is to adopt the concept of *syntactic semantics*. This concept was proposed by Rapaport (W.J. Rapaport 2002), who claims that the semantics is embedded inside syntax (i.e., the study of relations among symbols), and, hence, syntax can suffice for the semantic enterprise. To extend this concept to our domain, we claim that the semantics of a node is not determined by its label or name, instead can be represented by the role it plays in the network or its surrounding network structure. This idea is also conceptually similar to social position analysis (S. Wasserman and K. Faust 1994), which claims that it is the *role* a node plays in the

network, instead of its label, that determines its meaning. Using this idea, we propose to judge whether two nodes are similar by checking if they play similar roles in the network, and determine these roles, in turn, by analyzing the different combinations of surrounding nodes and links together with their network statistics. Consequently, two nodes that are physically far away from each other in the network can still be treated as similar given a similarity in their surrounding environments. We believe this syntactic semantic model for the meaning of the nodes is an adequate choice for an unsupervised system which tries to avoid using domain knowledge and introducing human biases.

The following sections described how we can model the role of the nodes in an MRN and identify abnormal nodes. We decompose the whole process into three stages. The first stage is structure modeling or feature selection. In this stage, the system *automatically* selects a set of features to represent the surrounding network structure of nodes. The second stage is a dependency computation or feature value generation stage. For this stage we design a set of different models to compute the dependency between the features and the nodes in the MRN. A *semantic profile* of a node is constructed by a set of features and their feature values. In the third stage the system tries to find the abnormal nodes by looking for those with abnormal semantic profiles.

2.1 Feature Selection Stage

To elucidate the feature selection stage, we start with a motivating example derived from Figure 1.2. There are four authors (A1, A2, A3, A4) in this MRN, and

let us assume our goal is to find out which author is most abnormal (i.e., plays different roles compared with others). After examining these nodes based on their connections (i.e., neighbor nodes and links), we can conclude several things about each of them:

A1 published two journal papers *(P1, P3)* and one of them cites the other. *A1* belongs to organization *O1*, has a colleague *A3*, and co-authored one paper with *A4*.

A2 published two journal papers *(P4, P5)* and one of them cites the other. *A2* belongs to organization *O2*, has a colleague *A3*, and co-authored one paper with *A4*.

A3 published one paper *P2* (no citation). *A3* belongs to two organizations *O1* and *O2*, and has two colleagues *A1* and *A2*.

A4 published two journal papers *(P3, P4),* one of them cites another paper and the other is cited by another paper. *A4* co-authored with two persons (*A1* and *A2*).

Based on the above description, it is not very hard to recognize that *A3* has the most abnormal semantics, since *A1* seems very similar to *A2*, and *A4* is still somewhat similar to both *A1* and *A2*. *A3* is not as similar to the others, since it published only one paper which has neither incoming nor outgoing citations. Unlike the others, it does not have any co-authors. Furthermore, it belongs to multiple organizations and others do not.

The above example shows that it is possible for humans to successfully identify abnormal nodes if we can somehow summarize the roles (or the surrounding structure) of them. However, our ultimate goal is to design an automatic mechanism to perform

this comparison. Therefore, we need a systematic method to model the semantics or roles of the nodes to make them comparable and contrastable automatically.

To realize this idea, we start by systematically listing the one and two-step paths of the four author nodes with their corresponding natural language interpretation (we omit one-step paths if they are already covered in a longer path):

A1:

A1—writes—P1—published_in—J1 (A1 writes P1 which is published in J1)

A1—writes—P3—published_in—J1 (A1 writes P3 which is published in J1)

A1—writes—P1—cites^{-1}*—P3* (A1 writes P1 which is cited by P3, note that *relation*$^{-1}$ stands for the inverse of *relation*)

A1—writes—P3—cites—P1 (A1 writes P3 which cites P1)

A1—writes—P3—writes^{-1}*—A4* (A1 writes P3 which is written by A4)

A1—belongs—O1—belongs^{-1} *–A3* (A1 belongs to O1 which is the organization of A3)

A2:

A2—writes—P4—published_in—J1 (A2 writes P4 which is published in J1)

A2—writes—P5—published_in—J2 (A2 writes P5 which is published in J2)

A2—writes—P4—cites^{-1}*—P5* (A1 writes P1 which is cited by P3)

A2—writes—P5—cites—P4 (A2 writes P5 which cites P4)

A2—writes—P4—writes^{-1}*—A4* (A2 writes P4 which is written by A4)

A2—belongs—O2—belongs^{-1} *–A3* (A2 belongs to O2 which is the organization of A3)

A3:

A3—writes—P2 (A3 writes P2)

A3—belongs—O1—belongs^{-1} *–A1* (A3 belongs to O1 which is the organization of A1)

A3—belongs—O2—belongs^{-1} *–A2* (A3 belongs to O2 which is the organization of A2)

A4:

A4—writes—P3—published_in—J1 (A4 writes P3 which is published in J1)

A4—writes—P4—published_in—J1 (A4 writes P4 which is published in J1)

A4—writes—P3—cites—P1 (A4 writes P3 which cites P1)
A4—writes—P4—cites^{-1}*—P5* (A4 writes P4 which cites P5)
A4—writes—P3—writes^{-1}*—A1* (A4 writes P3 which is written by A1)
A4—writes—P4—writes^{-1}*—A2* (A4 writes P4 which is written by A2)

In general one should be able to come up with descriptions of nodes similar to the ones we have provided in the motivating example by combining and maybe condensing the information provided by those paths. For example, in the case of A1, the first two paths tell us that A1 wrote two journal papers. The next two paths tell us that one paper cites the other. The fifth path reveals that A1 co-authored with A4 while the last path shows that A1 belongs to organization O1 and has a colleague.

This observation motivates a key idea of our approach: of using those paths to capture the semantics of the nodes. Another justification is that each path in a network can be translated into a standard logical notation by representing nodes as constants and links via binary predicates. Those predicates contain meanings and can be translated into natural language, as we did for the above paths. For example, in Figure 1.2 the path *A1—writes—P3—cites—P1* can be represented as the conjunction writes(A1,P3) $\wedge$ cites(P3,P1). This logical expression partly characterizes the meaning of the nodes A1, P1 and P3. It only partially characterizes meaning, of course, since there are many other paths (or logical expressions) that also involve these nodes. In our view, it is the combination of all paths a node participates in that determines its semantics. This differs from standard denotational semantics in which a node's interpretation is the object it denotes (R. K. Hill 1995).

Given these observations, one naïve approach to capture a *semantic profile* of a node is by treating all paths in the network as *binary features*, assigning true to the paths the given node participates in and false to the ones it does not. Thus, in a network of *k* different paths, each node can be represented by a *k*-dimensional binary feature vector. By doing this we have essentially transformed a multi-relational network into a propositional representation, where each node is a point in a high-dimensional space with attributes identifying its semantics based on its position and role in the network.

The previous paragraphs describe the central idea underlying the approach used for representing the semantic profile of a node, but there are still some problems that we must address. The first problem is that treating each path as a different feature generates an overfitting problem. Since each path is unique, the only nodes sharing a particular path feature would be those participating in the path, which would make these profiles useless for comparing nodes inside the path with the ones outside of it. For example, given the two paths cites(P2,P1) $\wedge$ published_in(P1,J1) and cites(P2,P1) $\wedge$ published_in(P1,J2), it might be important to compare and contrast J1 and J2. However, because these features would be considered independent, they could not really contribute to a meaningful comparison. A second problem relates to time and space complexity: a large semantic graph can easily contain millions of paths, and computation/storage in such high dimensional space would be costly.

These issues motivate the search for a more condensed feature set without losing the spirit of capturing the role semantics of instances. We do this by defining

equivalence classes between different paths that we call *path types* and then use these path types instead of individual paths as features. Whether two individual paths are considered to be of the same type will depend on one of several similarity measures we can choose. For example, we can view a set of paths as equivalent (or similar, or of the same type) if they use the same order of relations (called the *relation-only* constraint described in the Section 2.3). This view would consider the following three paths as equivalent:

cites(P2,P1) ∧ published_in(P1,J1)

cites(P2,P1) ∧ published_in(P1,J2)

cites(P2,P3) ∧ published_in(P3,J1)

Alternatively, we can consider two paths as equivalent if they go through the same nodes. This view would consider the following two paths as equivalent:

friends(Person1,Person2) ∧ direct(Person2,Movie1)

colleagues(Person1,Person2) ∧ act(Person2,Movie1)

Given these generalization strategies, then, the next question becomes how we can generate a meaningful and representative set of path types? One way is to apply a *variable relaxation* approach. Regarding the path cites(P2,P1) ∧ published_in(P1,J1) as an example, we find there are five ground elements in this path: cites, P1, P2, published_in and J1. If we relax one of its elements, say J1, to a variable ?X, then we get a new relaxed path type cites(P2,P1) ∧ published_in(P1,?X) which now represents a more general concept: “paper P2 cites paper P1 that is published in some journal”.

Further relaxing, we could also generalize a link such as published_in which would give us cites(P2,P1) $\wedge$?Y(P1,?X) or "paper P2 cites paper P1 that has something to do with some journal". In fact we can generalize any combination of nodes or links in a path to arrive at a more general path type. These path types still convey meaning but do so at a more abstract level. This makes them more useful as features to compare or contrast different instances or nodes. In Section 2.3, we will discuss a set of high-level constraints that allow our system to select path types automatically.

2.2 Feature Value Computation

A major advantage of using path types is that we do not limit ourselves to binary features (i.e., whether a node does or does not participate in a path). Instead, we can apply statistical methods to determine the dependence between a path and a node and use it as the corresponding feature value. This realizes one of the ideas proposed in the previous section, which is to employ statistical analyses of nodes and links as the basis of a similarity measure.

2.2.1 Performing Random Experiments on an MRN

General statistical correlation measures such as mutual information (MI) or pointwise mutual information (PMI) have been successfully applied to problems in a variety of areas. These measures rely on the existence of nondeterministic dependency between random variables. However, a normal multi-relational network, unlike a Bayesian network or general belief network, is a deterministic graph structure instead of a probabilistic one. It represents the relationships between nodes and normally there are no probabilities associated with these nodes and links.

Furthermore, the problem we face is different from, say, a typical natural language processing task in which one can learn the dependencies of words (i.e., the language model) from large corpora, since our task has no such corpora to take advantage of. As a result, questions such as "what is the MI between a node *x* and a node *y*" or "what is the PMI between a node *x* and a path type *p*" are ill-defined, because not only is there no uncertainty associated with *x* and *y*, neither are they random variables.

To apply statistical dependency measures to a deterministic MRN in order to compute the correlation between nodes and paths, we introduce a set of *random experiments* carried out on the MRN. A random experiment is an experiment, trial, or observation that can be repeated numerous times under the same conditions. The outcome of an individual random experiment has to be independent and identically distributed (i.e., it must not be affected by any previous outcome and cannot be predicted with certainty). Based on the results of these random experiments, we can create certain random variables and use them to measure the dependency between a node and a path type.

To elaborate this idea, we first introduce three simple random experiments to select a path in an MRN:

Random Experiment 1 (RE1): Randomly pick a path from the MRN. One can easily see that the probability of each path to be selected is 1/|Path|, where |Path| is the total number of paths in the MRN.

Random Experiment 2 (RE2): First randomly pick a node in the MRN, and then randomly pick a path that starts from the selected node.

Random Experiment 3 (RE3): First randomly pick a path type in the MRN, and then randomly pick a path among the ones that are of that path type.

Any of these three random experiments produces a single path as the output. However, the probability of each path to be selected varies for these three experiments, depending on the selection policy.

Based on the single path output of a random experiment, we can then introduce two random variables *S* and *PT*:

S: The starting node of this selected path

PT: The path type of this selected path

Note that both *S* and *PT* are discrete random variables, where the number of possible realizations in *S* equals the total number of nodes in the MRN and that of *PT* equals the total number of path types. Given these random variables, we can now compute dependencies between nodes and path types in a variety of ways which is described in the next sections.

2.2.2 Measuring Node/Path Dependence via Contribution

We start by an observation that each path type contains multiple realizations in the dataset. Take the path type writes(?X, ?Y) as an example: an instance A1 might occur in many paths of this type (say writes(A1, P1)…writes(A1,P99) representing that A1 wrote 99 papers), while another instance A2 might occur only in a few (say 1 time). Assuming that in the whole dataset only A1 and A2 write papers, we can say that A1 contributes 99%, A2 1% and the rest 0% to this "writing a paper" path type. That is, if a path is selected randomly with equal probability, then the conditional probability is

99% that given the path is of type writes, then the starting node is A1. In this case we can say that the dependency measure between the node A1 and path type writes is 0.99.

Associating this example to what we proposed in the previous section, it is not hard to see that here we have essentially computed a conditional probability of random variable *S* given *PT* based on RE1:

$$p_{RE1}(S = A1 \mid PT = writes)$$

We formalize this by defining the $contribution_k$ of an instance *s* to a path type p_t as the conditional probability that given a path with a certain path type p_t is randomly selected (based on Random Experiment *k*), this path starts from *s*:

$$contribution_k(s, p_t) = p_k(S=s|PT= p_t)$$

The contribution value therefore encodes the dependency information between a node and a path type. The concept is intuitive and understandable, which is a very useful and important characteristic for the problem of generating human-understandable explanations. Note that we consider *x* only as a starting point in a path and not at other positions. This is because if *x* is in the middle of a path, then it will be counted as the starting point of two shorter paths as exemplified below:

$$A \xrightarrow{R1} B \xrightarrow{R2} x \xrightarrow{R3} C = x \xrightarrow{R3} C + x \xrightarrow{(R2)^{-1}} B \xrightarrow{(R1)^{-1}} A$$

Similarly, if *x* is at the end of a path, then it is the starting node of the inverse path as well. Therefore, we consider only *x* as the starting point of paths to avoid redundancy.

Using path types as features and their contributions with respect to nodes as feature value, we can then construct what we call the *semantic profile* for a node. For example, here is an excerpt of the semantic profile (based on $contribution_1$) for the

director Ang Lee taken from a movie dataset used in some of the experiments described in the Chapter 5:

0.00105	[direct, has_actor, child_of]
0.00109	[direct, has_authors]
0.00111	[direct, has_actor, lived_with]
0.00161	[direct, remake]
0.00446	[write, has_cinematographer]
0.00794	[direct, has_actor, has_lover]

The first column represents the contribution_1 value and the second represents the relation-only path type. For example, the first row says that if one randomly picks a path representing "*x* directed a movie that has an actor who is the child of some person", than there is a 0.00105 probability that *x* is Ang Lee.

2.2.3 Measuring Node/Path Dependence via Mutual Information

As an alternative to the contribution measure, one can apply the concept of mutual information (MI) and pointwise mutual information (PMI) from information theory to compute the dependency between path type features and nodes.

2.2.4 Definition of MI and PMI

MI measures the mutual dependence of two random variables. The higher it is, the more dependent the two random variables are with each other. The equation of MI is as follows, where *X* and *Y* are two discrete random variables, *p(x), p(y)* represents the associated probability of their values and *p(x,y)* represent the joint probability (it can be proven that MI is always positive):

$$MI(X;Y) = \sum_{x,y} p(x,y) \log(\frac{p(x,y)}{p(x)p(y)}$$

For example, say a discrete random variable *X* represents visibility (i.e., good or bad) at a certain moment in time and random variable *Y* represents wind speed (i.e., high or low) at that moment. Then the mutual information between *X* and *Y* will be

$$MI(X;Y) = p(X = good, Y = high)\log(\frac{p(X = good, Y = high)}{p(X = good)\,p(Y = high)}) +$$
$$p(X = bad, Y = high)\log(\frac{p(X = bad, Y = high)}{p(X = bad)\,p(Y = high)}) +$$
$$p(X = good, Y = low)\log(\frac{p(X = good, Y = low)}{p(X = good)\,p(Y = low)}) +$$
$$p(X = bad, Y = low)\log(\frac{p(X = bad, Y = low)}{p(X = bad)\,p(Y = low)})$$

A high *MI(X;Y)* would imply that knowing the visibility tells a lot about the wind speed (and vice versa). In other words one can accurately predict the wind speed given the information about the visibility and vice versa. Conversely, low mutual information implies that knowing the value of one does not provide much information about the other.

Pointwise mutual information (PMI) measures the dependence between two *instances or realizations* of random variables instead of the random variables themselves:

$$PMI(X = x, Y = y) = \log(\frac{p(X = x, Y = y)}{p(X = x)\,p(Y = y)})$$

Note that unlike MI, PMI values can be negative. Negative PMI implies that two realizations are negatively correlated with each other. Zero PMI implies mutual independence between the two values. It is not hard to see that the mutual

information between two random variables is simply the integration of PMI values over all the pair-wise instances of these random variables.

Using the above example, we can measure the pointwise mutual information between two values of random variables *"X=good"* and *"Y=low"* as

$$\log\left(\frac{p(X = good, Y = low)}{p(X = good)\,p(Y = low)}\right)$$

High PMI implies that if we observe that visibility is good, then we are confident that the wind speed is low. Zero PMI implies that knowing visibility is good does not tell whether the wind speed is low and vice versa (independence). Negative PMI implies that if the visibility is good, then it is unlikely the wind speed is low.

2.2.5 Pointwise Mutual Information Model for Nodes and Paths in an MRN

With these two random variables, we can now model the dependency (i.e., feature value) of a node s and a path type p_t as the *pointwise* mutual information between $S=s$ and $PT=p_t$, which is $\log\left(\frac{p(S = s, PT = p_t)}{p(S = s)\,p(PT = p_t)}\right)$.

Note that the computation of the marginal probabilities and joint probability described in the above equation varies, depending on which random experiment one performs. For RE1 we get this following equation:

$$PMI_1(S=s, PT=p_t) = \log\left(\frac{\frac{|S=s, PT=p_t|}{|Path|}}{\frac{|S=s|}{|Path|} * \frac{|PT=p_t|}{|Path|}}\right)$$
$$= \log\left(\frac{|S=s, PT=p_t|}{|PT=p_t|}\right) - \log(|S=s|) + \log(|Path|)$$

$|S=s|$ is the total number of paths in the network that start from *s*. $|PT=p_t|$ is the total number of paths that have p_t as their path type. $|S=s, PT=p_t|$ is the total number of paths in the network that start from *s* and are of path type p_t.

For instance, suppose in a social network about movie people, there is a node *s* of the name "*Hitchcock*" and a relation-only path type p_t as *[has_friend, direct]* (this stands for "somebody has a friend who directed some movie"). Then, the PMI_1 value between "*S=Hitchcock"* and *"PT=[has_friend, direct]"* becomes:

$$\log\left(\frac{|S="Hitchcock", PT=[has\_frined, direct]|}{PT=[has\_frined, direct]}\right) - \log(|Hitchcock|) + \log(|Path|)$$

Note that $\frac{|S=s, PT=p_t|}{|PT=p_t|}$ represents the conditional probability "if a chosen path is of path type p_t, the probability it starts from *s*", which was defined as the *contribution₁ (s, p_t)* in the previous section. We can therefore rewrite *PMI_1* as:

$$PMI_1(S=s, PT=p_t) = \log(contribution_1(x, p_t)) - \log(|S=s|) + \log(|Path|)$$

From the above equation, we can find that *PMI_1* is positively correlated with the *contribution₁* between the node and the path type. The difference now is that there is an additional subtraction term *log*|*S*=*s*| and the positive *log*|*Path*|. The former alleviates the impact from nodes that are involved in many paths while the second is a

constant. This implies that compared with the contribution measure, the PMI_1 measure tries to discount the total number of connections for the node. In other words, if there are two nodes who contribute the same to a path type, the one that has fewer overall connections will have a higher PMI_1 value.

Below is the equation to calculate $PMI_2(S=s, PT=pt)$, which is the PMI values based on Random Experiment 2. It is different from PMI_1 since in Random Experiment 2 we first randomly selected a node and then a path type. Note that $|Node|$ represents the total number of nodes in the network.

$$PMI_2(S=s, PT=pt) = 1\log(\frac{\frac{1}{|Node|} * \frac{|S=s, PT=p_t|}{|S=s|}}{\frac{1}{|Node|} * \frac{\sum_{k \in node} \frac{|S=k, PT=p_t|}{|S=k|}}{|Node|}})$$

Here the term $\frac{|S=k, PT=p_t|}{|S=k|}$ stands for the probability that if from a source node k we randomly select a path, its path type is p_t. It is similar to the concept of conditional probability so we can rewrite it as $p(PT=p_t|S=k)$. Therefore, PMI_2 can be rewritten as the following equation:

$$\begin{aligned} PMI_2(S=s, PT=pt) &= \log(\frac{\frac{1}{|Node|} * p(PT=p_t|S=s)}{\frac{1}{|Node|} * \frac{\sum_{k \in node} p(PT=p_t|S=k)}{|Node|}}) \\ &= \log(\frac{p(PT=p_t|S=s)}{\sum_{k \in node} p(PT=p_t|S=k)}) + \log(|Node|) \end{aligned}$$

The interpretation of PMI_2 turns out to be less complicated than it looks. It captures the dependency between a node and a path type based on the "node's point of view". It conveys the idea that the PMI_2 between a node *s* and a path type p_t will be high if p_t is a very frequent path type starting from *s*. For instance, in our previous example the PMI_2 will be high between *s=Hitchcock* and p_t*=[has_friend, directed]* if most of the paths describing Hitchcock in the dataset are about some of his friends that directed some movie.

Similar to the previous PMI values, PMI_3 tries to model the dependency of a node and a path type in a deterministic network based on a path-choosing experiment. The difference lies in the way the path is picked. The calculation of PMI_3 is as follows, where |*Path Type*| stands for the total number of path types in the network:

$$PMI_3(S=s, PT=pt) = \log\left(\frac{\frac{1}{|PathType|} * \frac{|S=s, PT=p_t|}{|PT=p_t|}}{\frac{1}{|PathType|} * \frac{\sum_{m \in path\_type} \frac{|S=s, PT=m|}{|PT=m|}}{|PathType|}}\right)$$

Again, the term $\frac{|S=s, PT=m|}{|PT=m|}$ stands for the probability that if we randomly pick a path of path type *m*, the chance that this path starts from *s*. Therefore we represent it as *p(S=s| PT=m)*, and rewrite PMI_3 as:

$$PMI_3(S=s, PT=pt) = \log\left(\frac{p(S=s|PT=p_t)}{\sum_{m \in path\_type} p(S=s|PT=m)}\right) + \log(|PathType|)$$

Similar to PMI_2, there is an intuitive interpretation behind this equation. PMI_3 measures the dependency of a node and a path type based on the "path's point of view" (while PMI_2 is based on the "node's point of view"). It says the PMI value between a node *s* and a path type p_t will be high if among all the possible nodes, *s* has a very high chance to be the starting node of p_t. For instance, in our example *s=Hitchcock* and p_t *=[has_friend, directed]* will have high PMI_3 value given Hitchcock is the only person who has some friend who directed some movie.

2.2.6 Mutual Information Model for Multi-Relational Networks

Unlike PMI, which models the dependency of two *instances* of random variables, mutual information is generally used to compute the dependency of two random variables. Since our task focuses on modeling the *instances* in the network, it seems to be more natural to apply the concept of PMI as discussed in the previous section.

However, as can be inferred from the examples stated above, the PMI models we proposed focus only on the positive dependency (e.g. if *s* happens, whether p_t happens and vice versa) and ignore other situations (e.g. if *s* does not happen, whether p_t does or doesn't happen). Ideally, we would like to consider dependency from all perspectives, which is something that can be achieved by using full mutual information analysis. Based on the same path-choosing random experiments, our design of the mutual information dependency model starts by redefining *S* and *PT* as two *binary* random variables:

S: whether the path starts from the node *s*

PT: whether the path belongs to the path type p_t

Since both *S* and *PT* are binary random variables (i.e., their values can be either true or false), we can compute the mutual information between them as follows:

$$MI(S,PT)= \quad p(S=t,PT=t)*\log\frac{p(S=t,PT=t)}{p(S=t)*p(PT=t)}+$$
$$p(S=f,PT=t)*\log\frac{p(S=f,PT=t)}{p(S=f)*p(PT=t)}+$$
$$p(S=t,PT=f)*\log\frac{p(S=t,PT=f)}{p(S=t)*p(PT=f)}+$$
$$p(S=f,PT=f)*\log\frac{p(S=f,PT=f)}{P(S=f)*p(PT=f)}$$

Again, the formulas for the joint and marginal probabilities vary for different random experiments, depending on how a path is selected. Note that the major difference of these two random variables compared with the previous ones is that both *s* and p_t are *included in* the definition of random variables and their values can only be true of false — using the PMI analysis, in contrast, both *s* and p_t are *instances* of random variables and they do not show up in the definition of random variables. The MI measure considers all the possible combinations of positive and negative instances, and consequently reveals how confident one can be in terms of predicting whether a path type is p_t given we know whether the starting node of a randomly selected path is *s*, and vice versa (i.e., predict the starting node based on the path type).

2.3 Meta-constraints

We have described that path types can be generated from paths by a method we call variable relaxation. Given that every element in a path can be relaxed to a variable, then, how can the system systematically select a set of path types and how

can it adapt to users' preferences when necessary? In this section we design several meta-constraints as parameters that can guide the system to select a set of path types automatically. Moreover, we also allow freedom for users to introduce domain knowledge or biases through such high-level constraints if necessary.

Meta-constraint 1: maximum path length

The system needs to limit the path length while selecting path types as features. Constraining the path length is reasonable in the sense that the farther away a node/link is to the source node, the less impact it has on the semantics of the source node. Moreover, it is not hard to image that the longer a path is, the harder it is for humans to make sense of it. Therefore, for explanatory purposes, we would like to avoid long paths. In our experiment we chose the maximum path length somewhere between four and six. The choice is based on the size of the search space and how far away one believes link information to still have influence on the meaning of the source.

Meta-constraint 2: relation-only constraint

This constraint tells the system to treat paths with the same sequence of relations (links) as of the same path type. In other words, it considers only the link types and ignores any information provided by the nodes. We choose this as the default constraint to create the semantic profiles, since in general we assume that the typed links play a more important role and can convey more information than the labels of nodes in a semantic graph.

Based on meta-constraints 1 and 2, the system can fully automatically extract a set of path types from an MRN to represent the semantics of the nodes.

Meta-constraint 3: node and link type constraints

This type of constraint allows users to express their preference in terms of the types of nodes and links that should be considered. For example, one can specify that at least one of the nodes in a path type needs to be of type *person,* or that one link in the path needs to be a *murder* link.

Meta Constraint 4: exclusion constraint

This constraint allows the user to say what should not be considered. For example, one can state that paths that contain the *rob* relationship should not be considered in the analysis.

By combining meta-constraints 3 and 4, users can express their preference and biases. This is particularly useful in situations where users are very confident about what kind of links or nodes are important and what are useless. Some of those meta-constraints might have been implicitly considered already during the construction phase of the network.

Meta-constraint 5: structure constraints

This type of constraint controls the structure of a path. For example, in one of our experiments we ask the system to only consider paths whose source and target nodes are the same, which we call the “loop constraint”. We also define a “uniform link constraint” to consider only paths with only one single link type such as [A cites B cites C cites D].

Meta-constraint 6: guiding the computation of feature values

These types of constraints are used to guide the process of feature value selection by constraining the random experiments performed. Meta-constraints 3, 4, and 5 can be reused again here. For example, we can tell the system to ignore all paths without a certain type of nodes while performing the path-choosing random experiments.

2.4 Finding Abnormal Nodes in the MRN

Using path types as features and the dependencies (i.e., contribution or PMI or MI) of each node with respect to each path type as feature values, we now have a method for representing the semantic profiles of nodes in a propositional attribute-value form. Each node profile is represented by a numeric feature vector that records the dependency of the node with each path type in the dataset.

Now that we have a metric for ascertaining the semantic profile of a node, then, the next step is to identify abnormal instances in our set of nodes. One approach is to extract nodes that have high dependency values for many path types (e.g., highly connected nodes would fall into this category). The opposite approach would be to identify the ones that have low dependency values such as isolated or boundary nodes.

Although these two types of nodes are potentially interesting, we believe they are not extremely desirable candidates for our system to find. In fact unlike centrality theory or PageRank, our goal is not to find important nodes, but instead we are trying to find nodes that look *relatively different* from others. In particular, we are

interested in nodes whose semantic profiles are significantly different from those of other nodes.

We transform the question of *identifying abnormal nodes in a semantic graph* into the question *identifying nodes with abnormal semantic profiles* because mining propositional data is a well-studied problem in data mining that has a wide range of techniques available. For example, below we will introduce a variety of outlier detection techniques that can assist us to identify abnormal nodes.

An outlier is an observation that deviates so much from other observations to arouse suspicion that it was generated by a different mechanism (D. Hawkins 1980). There are three major classes of outliers: clustering-based, distribution-based and distance-based outliers. Each has its associated detection algorithms. Clustering-based outlier detectors such as CLARANS (R.T. Ng and J. Han 1994), BIRCH (T. Zhang, R. Ramakrishnan et al. 1996) and CLUE (S. Guha, R. Rastogi et al. 1998) extract outliers as those points that cannot be clustered into any clusters based on a given clustering method. Distribution-based outlier detection assumes some statistical distribution of the data and identifies deviating points as outliers (V. Barnett and T. Lewis 1994). We argue that in terms of extracting abnormal nodes based on our semantic profiles, neither of these is a good option. This is because our system is intended to deal with data in arbitrary domains, and there is no guarantee that either a learnable distribution or distinguishable clusters exist. Distance-based outlier detection, on the other hand, is more generic and applicable to our problem. It identifies outlier points simply as those that look very different from their neighbor

points (E. Knorr and R. Ng 1998). Consequently, it does not rely on the existence of clusters or learnable distributions. Distance-based outlier detectors look for outliers from a *local* point of view instead of a *global* one. That is, they do not look for a point that is significantly different from *the rest of the world*, but for a point that is significantly different from its *closest points*. For example, a distribution-based outlier detector might not deem a researcher who published three papers per year as very abnormal, while a distance-based outlier detector can identify it as an abnormal one, if finds that the other researchers in the same area (i.e., the neighbor points) published on the average one paper every two years. This strategy makes intuitive sense because we do not usually compare the productivity of researchers from different fields. A distance-based outlier method allows us to identify nodes that are different from other nodes *in the same context*, and we believe these types of abnormal instances have a higher chance to interest users. The examples in Chapter 1 also indicate the usefulness of distance-based outliers in a security domain, since malicious individuals who try to disguise themselves by playing certain roles but fail to get everything right are likely to still be different from genuine players of that role.

In the experiments we conducted, we chose Ramaswamy's distance-based outlier algorithm (S. Ramaswamy, R. Rastogi et al. 2000), which ranks outlier points by their distance to the *k*-th nearest neighborhood. Outliers are thus considered points that are far away from their *k* closest neighbors.

2.5 UNICORN: An Unsupervised Abnormal Node Discovery Framework

Based on the methodologies and techniques discussed above, we can now describe how our node discovery framework UNICORN[1] identifies abnormal nodes in a multi-relational network. The information flow is shown in the upper part of Figure 2.1. First, the system creates a set of path types to use as features in the semantic profiles of nodes. It is important to note that this set of features is not defined by the user but instead computed directly from the data using the very general meta-constraints described previously. For example, in one of our experiments we tell the system to find all different relation sequences (a relation-only constraint) up to length four in the data and use those as features. Once the set of path types is determined, UNICORN then computes the dependency (i.e., either contribution, MI or PMI based on one of random experiments) for each node and path type as the feature values to generate a semantic profile for each node. Finally, UNICORN applies an outlier detector or outlier ranking algorithm to find nodes with abnormal semantic profiles. The lower part of Figure 2.1 describes a system that produces the explanations for UNICORN, will be elaborated Chapter 4.

[1]UNICORN, as an abnormal animal, is the abbreviation for “UNsupervised Iinstance disCOvery in multi-Relational Networks”.

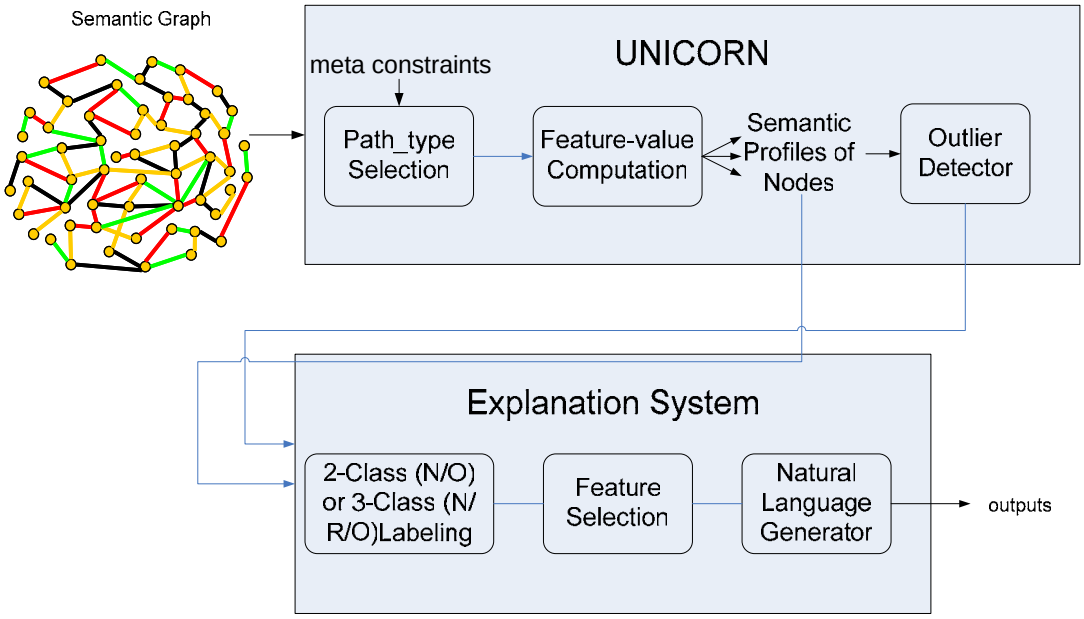


Figure 2.1 Information flow in UNICORN

The following pseudo-code describes UNICORN in more details. Note that for simplicity of description, *extract_path_types* implicitly assumes the "relation-only" view in the way it generates candidates, and we choose contribution$_1$ as feature values. It utilizes breadth-first search to find all relevant path types. Also, the output of function *get_outliers* can be either a set or a ranked list of nodes depending on the kind of outlier algorithm that is applied. Given an outlier *detection* algorithm, *O* will be a set of outlier nodes. If it is an outlier *ranking* algorithm, the result will be a ranked list of nodes corresponding to their degree of "outlierness".

function *UNICORN (G, mc, k) {*

// Input *G* is an MRN *<N, R, L>* with nodes $N=\{n_1,n_2 \ldots n_{|N|}\}$, edges (or relations) //$R=\{r_1,r_2 \ldots r_{|R|}\}$ and link types $L=\{l_1,l_2,\ldots l_{|L|}\}$. Each r_i links a source node *s* to a target node *t* //via a link of type l_j.

// Input *mc* is a meta constraint for selecting path types (e.g.,"relation-only")

// Input *k* is the maximum path length considered in the path type computation.

1. P_t := *extract_path_types(G, mc, k)*
2. ***array*** *profile[|N|, $|P_t|$]*
3. ***for*** *n := 1* ***to*** *|N|*
4. ***for*** $p_t \in P_t$
5. *profile[n, p_t] := get_contribution(G, n, p_t)*
6. ***return*** *get_outliers(profile)*

function *extract_path_types(G, mc, k)*

1. *pathLength := 0*
2. $P_t := \{l_1, l_2, \ldots l_{|L|}\}$
3. ***for*** *pathLength := 1* ***to*** *k-1*
4. ***for*** $p_t \in P_t$ *where length(pt) = pathLength*
5. ***for*** $l \in L$
6. p_t*' := concatenate (p_t, l) // add link l to the end of* p_t
7. ***if*** *path_exists(G,* p_t*')* ***and*** *satisfies(*p_t*', mc)*
8. $P_t := Pt \cup p_t$'
9. ***return*** P_t

function *get_contribution(G, s,* p_t*)*

1. *starts := number of paths of type* p_t *in G that start with s*
2. *all := number of paths of type* p_t *in G*
3. ***return*** *starts / all*

An important aspect of UNICORN is that it is designed primarily as a framework with a variety of options at each stage that users can choose from. In the feature selection stage, our meta-constraints provide users a certain amount of

flexibility to introduce their preferences, without affecting the overall advantage of being domain independent. In the feature value generation stage, users can choose not only three different random experiments (RE1 as default) but also several plausible dependency models: contribution (default), PMI measures and MI measures, each of which has a slightly different view and intuition for node/path dependency. In the final stage, users can plug in different types of outlier detection or outlier ranking algorithms to find different types of abnormal instances. Later in Chapter 6 we will describe a novel outlier detection algorithm we will call the "explanation-based outlier detector" which can also be used in this stage.

2.6 Local Node Discovery

The process can be easily adapted to solve a modified version of the problem, which we call the *local node discovery* problem. Its goal is to identify a node that is abnormally connected to a given node *s*. For example, given a medical MRN created based on a set of diseases, foods, and patients, one might want to see if there is a certain food that is abnormally connected to a given disease. Such a food could be interesting, because it might indicate the cure or cause of the disease. To perform such discovery, we need a system that can takes a given source node as input and that outputs nodes that are abnormally connected to it. We need only to modify UNICORN slightly to address this task. The difference is that when computing the feature values for each path type, we consider only the paths that start from *s*. This can be done by simply adding one meta-constraint of type 6 during the feature value generation stage. In this case the dependency measures will be focused on picking

only paths starting from *s*. For example, the contribution of a node *n* with respect to a path type p_t is the total number of times a path of type p_t starts from *s* and ends in *n*, divided by the total amount of times a path of type p_t starts from *s*. The justification for this is that we are looking for nodes that are abnormally connected with *s*, thus we care only about paths that include *s*.

Chapter 3

Evaluating UNICORN

In this chapter we describe our experiments based on an organized crime dataset to evaluate UNICORN. The goal of this evaluation is to demonstrate the usefulness of the system by showing that it can identify suspicious crime participants, and that it does much better than other state-of-the-art network algorithms that have been used for analysis of the 9/11 terrorist network. Note that the goal of this evaluation is not to show that the instances found by UNICORN are truly abnormal. We leave this part to the explanation system in the sense that the abnormality of the discovered instances can be validated by showing why they are different from the others in a human understandable form.

3.1 The Dataset

We evaluate our system by applying it to a synthetic dataset in the domain of Russian organized crime. Our main motivation for using a synthetic dataset is that this type of dataset has an answer key describing which entities are the targets that need to be found. In this experiment we evaluate how well our algorithm can identify the predefined suspicious instances given in the answer key of the dataset. We also compare our algorithm with a selection of state-of-the-art unsupervised network algorithms such as PageRank, HITS, social network analysis (centrality theory), etc.

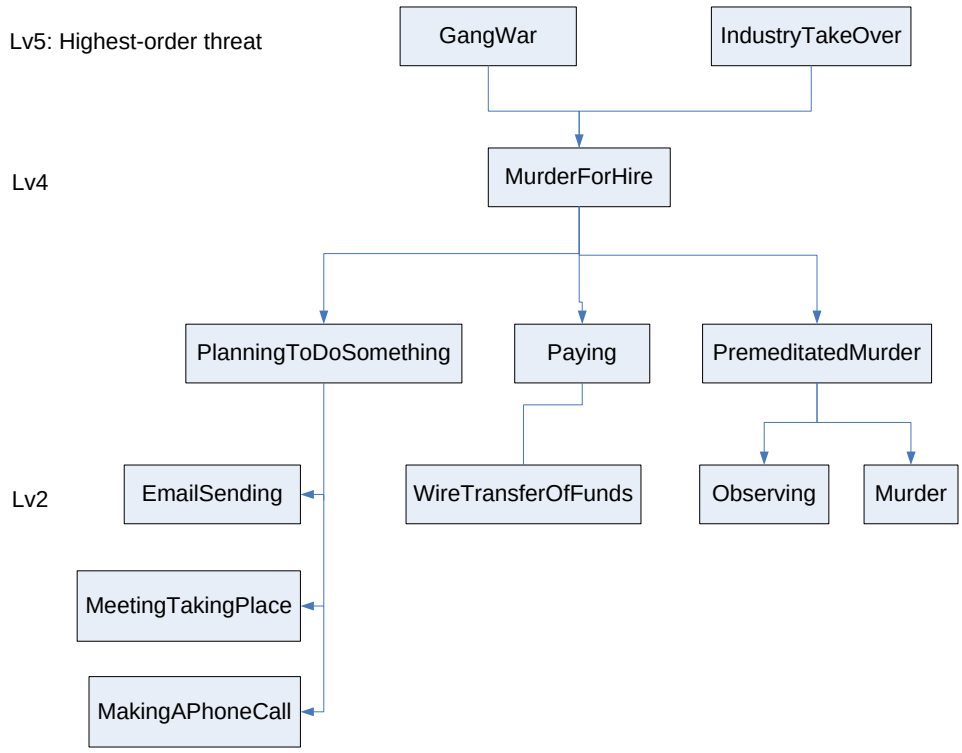


Figure 3.1: Event type hierarchy of the Russian organized crime dataset

The data we used came from a suite of simulated datasets designed by Information Extraction & Transport, Inc. as part of DARPA's Evidence Extraction and Link Discovery program. The data simulates a Russian organized crime (or Mafiya with a "y") domain with a large number of entities involved in activities such as contract murders, gang wars and industry takeovers. For each dataset we are given an answer key containing the information of participants in high-level events of interest such as GangWar and IndustryTakeOver. Those participants are not explicitly described in the data but need to be inferred from lower-level, incomplete and noisy evidence. With these answer keys we can test our program by checking if the abnormal nodes it discovered match the hidden higher-level crime participants.

Figure 3.1 illustrates the event hierarchy of the dataset. There are two different types of top-level events: gang wars and industry takeovers. Gang wars occur between two rivaling Mafiyas, and industry takeovers are attempts by one Mafiya to take over an industry. Both events are composed of various levels of lower level events such as murders for hire (or contract murders), communications, financial transactions, etc.

We perform this experiment by applying our system to find the most suspicious Mafiyas. UNICORN first generates the semantic profile for all Mafiyas, and then applies Ramaswamy's distance-based outlier algorithm to rank them. Our goal is to see if the top abnormal Mafiyas our system identified match the high-level crime organizers described in the answer key. We performed a similar experiment for the set of industries, where we want to know if the most abnormal industry the system found is the one being taken over.

Table 3.1: The statistics of the six datasets

Data	Size	Observ-ability	Noise	# of nodes	# of links	# of node types	# of link types	# of Mafiyas	# of Indus-tries
D1	large	low	2	9429	16257	16	31	42	21
D2	large	average	1	8418	12897				
D3	large	high	0	6346	8349				
D4	medium	low	2	9142	15425				
D5	medium	average	1	8033	12687				
D6	medium	high	0	6172	7848				

We tested on 6 different datasets (D1 to D6) whose characteristics are described in Table 3.1. Each dataset contains roughly 6000-9000 nodes while the number of

links ranges from 8000-16000. There are 16 different node types representing objects and events, which are:

Industry, BankAccount, Person, Business, Mafiya, TelNumber, MakingAPhoneCall, MeetingTakingPlace, PlanningToDoSomething, WireTransferOfFunds, Paying, EmailSending, Observing, Murder, PremeditatedMurder, MurderForHire

There are 31 different link types representing the relationships between those nodes including (detailed descriptions of these relations are listed in Appendix I):

accountHolder, callerNnumber, ceo, dateOfEvent, deliberateActors, deviceUsed, employees, eventOccursAt, geographicalSubregions, hasMembers, mediator, murderer, objectsObserved, operatesInRegion, orderedContractMurder, orgKiller, orgMiddleman, payee, payer, perpetrator, phoneNumber, receiverNumber, recipient, relatives, sender, socialParticipants, subevents, transferMoneyFrom, transferMoneyTo, victim, vor

The data sets differ in size (large or medium), observability (low, average, or high) and noise (0, 1, 2). The size parameter does not necessary reflect the total number of nodes and links in the data set. Instead it represents how many contract murder events are in the data. The large-sized data sets contain 20 contract murders while the medium-sized ones have only 10. Observability is the measure of evidence that is explicitly revealed. In other words, the lower the observability, the less evidence of events that is reported in the data. The noise parameter represents the degree of noise in Lv2 events. Noise level 2 datasets have on average 2000 noise events, noise level 1 datasets have 1000 and there are no noise events in noise level 0 data sets. Note that the difficulty of the datasets can be ranked as follows: {D1,D4}>{D2,D5}>{D3,D6}. D1 and D4 possess the lowest observability and the

highest noise. D3 and D6 have the highest observability and lowest noise, therefore should be simpler to process.

3.2 Comparing Different Algorithms

We compare our results with other network ranking algorithms including PageRank (L. Page, S. Brin et al. 1998), HITS (J.M. Kleinberg 1999) and Betweeness Centrality (S. Wasserman and K. Faust 1994), which have been used to analyze the 9/11 terrorist networks (V. Krebs 2001; J. Qin, J. Xu et al. 2005). PageRank ranks the importance of a node by the quality of the links pointed to the node. That is, a node will have high PageRank value if it is pointed at by a lot of nodes that themselves have high PageRank values. HITS (hyperlinked induced topic search) considers both incoming and outgoing links for each node and generates two scores called hub and authority values. These two values have mutually reinforcement one another. An authority value is computed as the sum of the hub values of the pages that point to that page. A hub value is the sum of the authority values of the pages it points to. In our experiment we rank the nodes based on the authority values. Betweenness is a centrality measure of a node in a graph. Nodes that occur on many shortest paths between other nodes have higher betweenness than those that do not. In the experiments we treat all the typed links equally (i.e., as a single-relational network) for these three algorithms, since they do not distinguish link types.

Table 3.2: Ranking of target nodes in the organized crime dataset. For each dataset, there are two or four targets to be found. The numbers show how each algorithm ranks those targets.

Data	Results				
	To Be Found	UNICORN	HITS	Betweeness	PageRank
D1	Mafiya_gang1	***2***	15	41	41
	Mafiya_gang2	***3***	5	32	39
	Mafiya_takeover	***1***	***1***	***1***	22
	Industry	***1***	6	18	17
D2	Mafiya_gang1	***1***	***3***	10	31
	Mafiya_gang2	***2***	***2***	***2***	37
	Mafiya_takeover	***3***	***1***	28	18
	Industry	***1***	12	2	3
D3	Mafiya_gang1	***1***	***2***	***2***	38
	Mafiya_gang2	***2***	***1***	9	40
	Mafiya_takeover	4	6	12	41
	Industry	***1***	4	***1***	***1***
D4	Mafiya_takeover	2	2	10	34
	Industry	***1***	18	39	19
D5	Mafiya_takeover	3	5	19	42
	Industry	***1***	15	17	21
D6	Mafiya_takeover	***1***	***1***	***1***	42
	Industry	***1***	***1***	14	21

Results of the experiment are described in Table 3.2. The second column lists the targets to be found for each dataset. Large datasets have a gang war (which includes two Mafiyas: Mafiya_gang1 and Mafiya_gang2 fighting with each other) and an industry takeover event (which contains one Mafiya: Mafiya_takevoer trying to take over an industry), while medium datasets only have one industry takeover event. The numbers in the last four columns represent the ranking of those targets based on different algorithms. For example, for dataset D1 UNICORN ranks Mafiya_gang1 as the second in its list of abnormal individuals, while HITS rank it as the 15th. In the large-size dataset, a perfect algorithm should rank the three target Mafiyas as the top three and the target industry as the top one. In medium-sized datasets such as D4 to

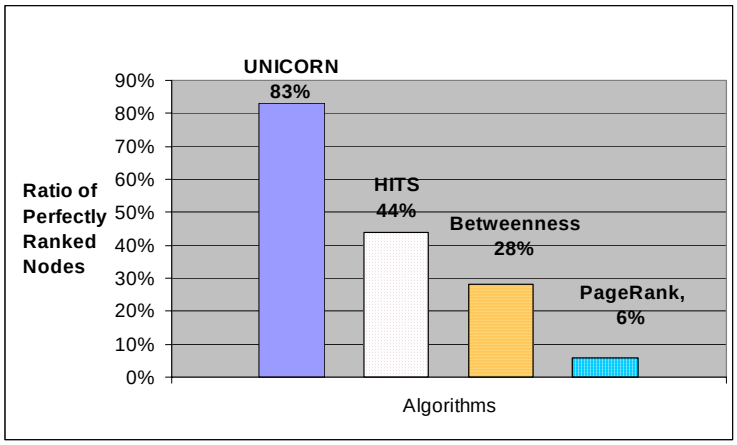
D6 which contain only one IndustryTakeOver event, a perfect algorithm should rank both the Mafiya_takeover and target industry as the top one. In all the experiments reported in this chapter, UNICORN utilizes its default relation-only meta-constraint to choose path types. In the first experiment shown in Table 3.2, the contribution$_1$ dependency measure was applied and the maximum path length was set to 4.

The results in Table 3.2 show that the threat instances that are supposed to be found in the answer key are consistently ranked on top of UNICORN's list, while this is not the case for the other three algorithms. We call a crime participant as ranked *perfectly* by an algorithm if there is no other innocent candidate ranked higher than it. Figure 3.2(a) shows that overall 83% (15 out of 18 in six datasets) of the crime participants are ranked perfectly by UNICORN, while the second-best algorithm HITS can rank only 44% (8 out of 18) of them perfectly.

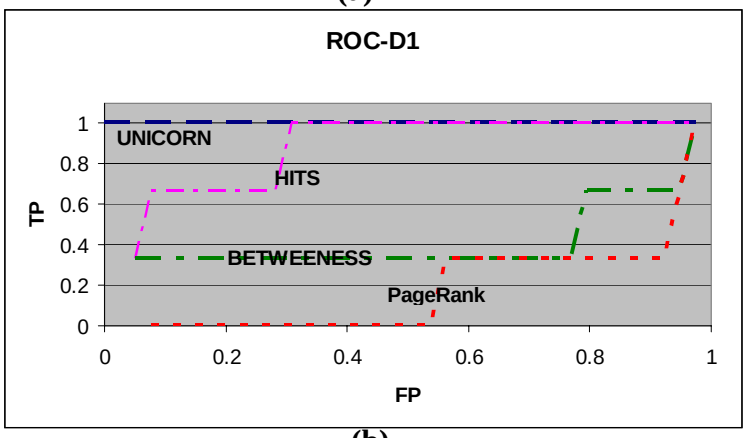
In fact, this measurement is the same as the average precision and recall values if we consider the top three ranked nodes are labeled as "positives" and the rest as "negatives". Precision and recall are identical in this scenario because the number of positively labeled nodes (3) is the same as the number of nodes that are supposed to be found. Alternatively, we can compute the Receiver Operating Characteristics (ROC) curve for the results. ROC curves plot the false positive rate FP as the *X*-axis and true positive rate (TP) as the *Y*-axis while sweeping the classification threshold. A good algorithm should have high TP while FP is till low. The reason to use ROC curve is that it is not sensitive to arbitrary threshold while the quality of ranking can be mapped to the measure of the area below the curve. Figure 3.2(b) is the ROC

curve for the results from each algorithm on D1. The result is consistent with 3.2(a) as UNICORN performs much better (reaches 100% TP while FP is 0%) than the second best algorithm HITS by a decent margin. Neither Betweeness Centrality nor PageRank seem to be very useful for this problem.

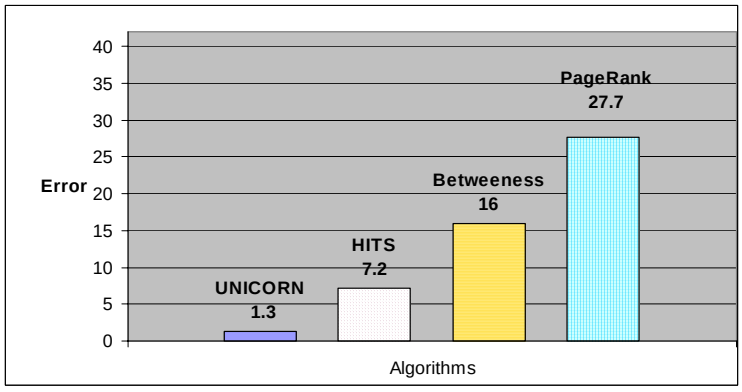
We also compare the error of different algorithms. To measure the error, we compute on average how many candidates that are *not* in the answer key are ranked higher than the non-perfectly ranked crime participants (or, to look at the problem another way, the average number of spots that a non-perfectly ranked crime participant is away from its ideal position). For example, there are three target nodes not ranked perfectly by UNICORN: Mafiya_takeover in D3 (ranked fourth), Mafiya_takeover in D4 (ranked second), and Mafiya_takeover in D5 (ranked third). For the case of D3 and D4, there is one innocent Mafiya group ranked higher than the target one. In the case of D5, there are two other innocent Mafiya groups ranked higher because the target one is ranked third. As a result the average error is (1+1+2)/3=1.3. This value denotes that these Mafiyas are on average 1.3 spots away from their ideal ranking. It is obvious that the lower the error the better. Figure 3.2 (c) demonstrates that the error of our system (1.3) is much better than the second-best HITS algorithm, in which each non-perfectly ranked target is 7.2 spots away from its ideal ranking. Both betweeness centrality and PageRank did far worse jobs in this task in both accuracy and error measurements.



(a)



(b)



(c)

Figure 3.2: Evaluation of UNICORN: (a) shows how many of the hidden crime organizers can be found by each method. (b) illustrates the ROC-curve for dataset D1. (c) shows on average how many innocent individuals are ranked higher than the target individuals

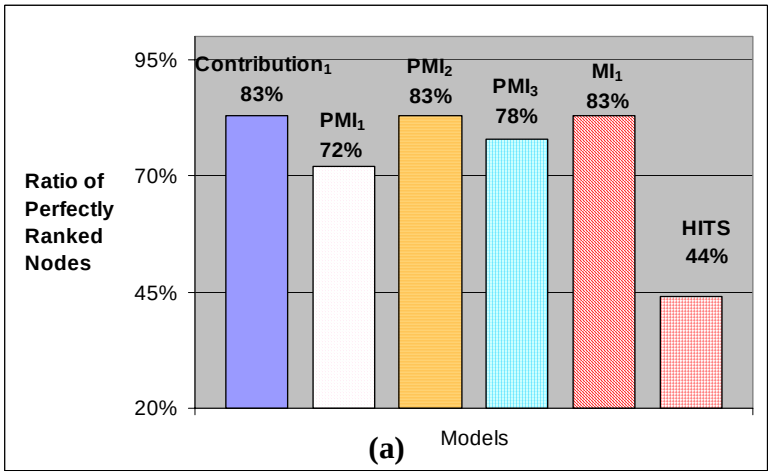
We contend that the primary reasons for UNICORN's exceedingly high performance compared with the other algorithms are twofold: First, it has the capability to utilize information provided by the different links types while the other algorithms do not take the semantics of links into account. Second, for crime analyses, using a distance-based outlier method (i.e., finding nodes that are different from their neighbors) seems to be a better option compared with using centrality or importance.

3.3 Comparing Different Dependency Models

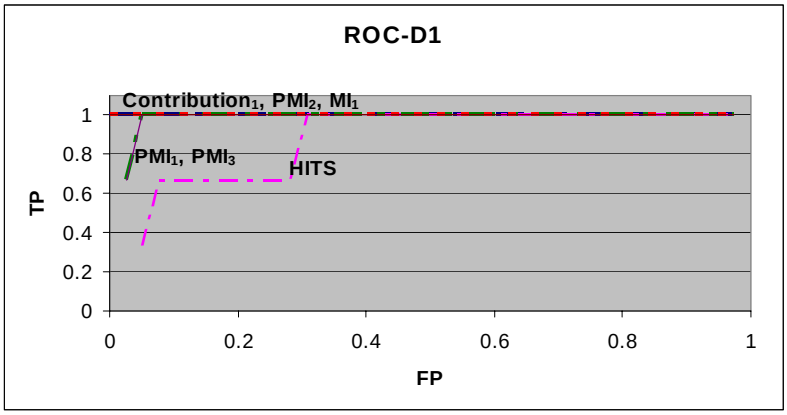
The next table compares the performance of several different dependency measures that can be used by UNICORN (note that for the contribution and MI measure, this experiment used only RE1, in which every path has the same probability of being chosen). The results are shown in Table3.3 and Figure 3.3.

Table 3.3: Comparing different dependency measures for UNICORN

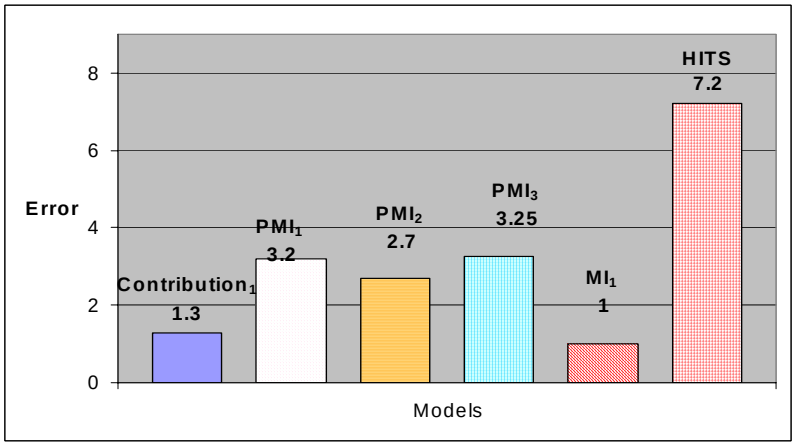
Data	Results					
	To Be Found	*Contribution*$_1$	PMI$_1$	PMI$_2$	PMI$_3$	MI$_1$
D1	Mafiya_gang1	***2***	***1***	***1***	4	***1***
	Mafiya_gang2	***3***	***2***	***2***	***2***	***2***
	Mafiya_takeover	***1***	4	***3***	***1***	***3***
	industry	***1***	***1***	***1***	***1***	***1***
D2	Mafiya_gang1	***1***	***1***	***1***	***1***	4
	Mafiya_gang2	***2***	***2***	***2***	***2***	***2***
	Mafiya_takeover	***3***	***3***	***3***	***3***	***1***
	industry	***1***	***1***	***1***	***1***	***1***
D3	Mafiya_gang1	***1***	***1***	***1***	***1***	***1***
	Mafiya_gang2	***2***	***2***	***3***	***2***	***2***
	Mafiya_takeover	4	***3***	***2***	***3***	***3***
	industry	***1***	***1***	***1***	***1***	***1***
D4	Mafiya_takeover	*2*	6	5	4	2
	industry	***1***	***1***	2	***1***	***1***
D5	Mafiya_takeover	*3*	8	4	8	2
	industry	***1***	3	***1***	3	***1***
D6	Mafiya_takeover	***1***	2	***1***	***1***	***1***
	industry	***1***	***1***	***1***	***1***	***1***



(a)



(b)



(c)

Figure 3.3: Comparing different dependency measures for UNICORN

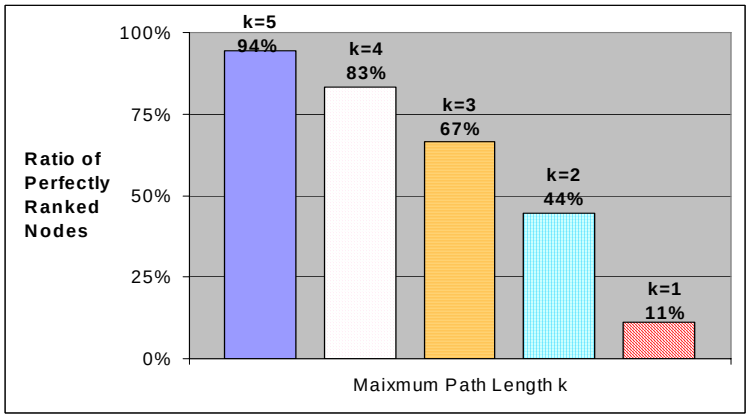
Again, we chose the relation-only constraint with path length up to 4. The results demonstrate that all the dependency models can reach a fairly good accuracy. About 70~80% of the crime participants received perfect rankings while the rankings for the rest are still very close to the top (about 1 to 3 spots away). The ROC curve shows that contribution$_1$, PMI$_2$ and MI$_1$ perform perfectly on D1, while the PMI$_1$ and PMI$_3$ are not too far away. All models perform significantly better than the second best algorithm HITS. MI is the best overall because of the smallest error (though not by a significant margin), which shows that using information from both positive and negative samples is, in this case, a more accurate measure.

3.4 Comparing Different Path Lengths

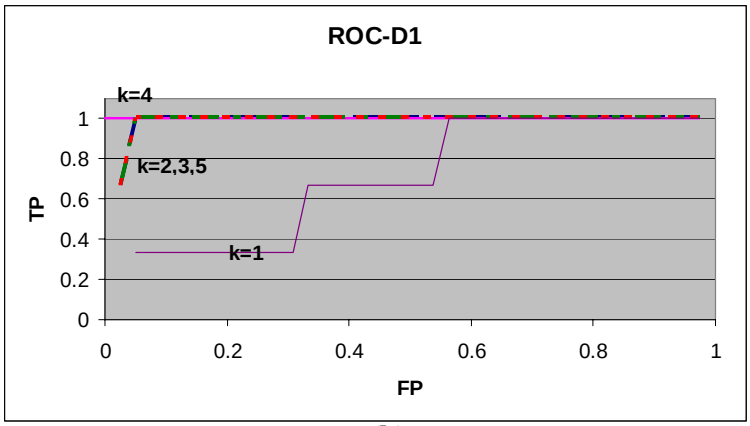
Table 3.4 and Figure 3.4 compare different path length constraints (k=1 to 5).

Table 3.4: Comparing different path lengths for UNICORN

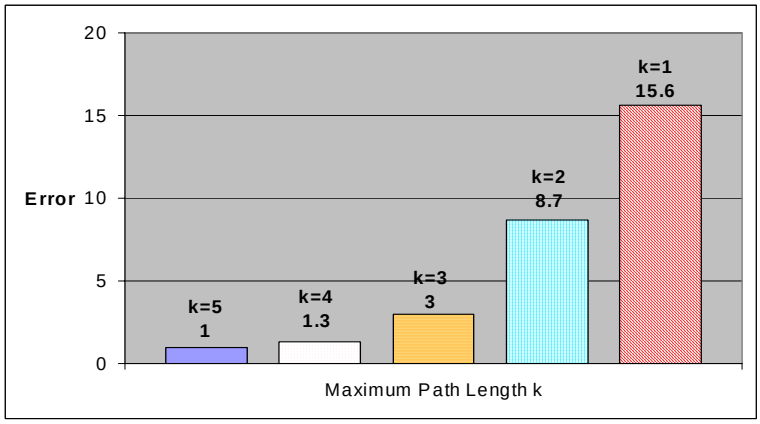
Data	Results					
	To Be Found	***k=5***	***k=4***	**k=3**	**k=2**	**k=1**
D1	Mafiya_gang1	***4***	***2***	***1***	***1***	24
	Mafiya_gang2	***2***	***3***	5	4	3
	Mafiya_takeover	***1***	***1***	3	3	15
	industry	***1***	***1***	***1***	10	11
D2	Mafiya_gang1	***1***	***1***	***1***	***1***	15
	Mafiya_gang2	***2***	***2***	***2***	***2***	32
	Mafiya_takeover	***3***	***3***	***3***	***3***	29
	industry	***1***	***1***	***1***	10	11
D3	Mafiya_gang1	***1***	***1***	***1***	***1***	5
	Mafiya_gang2	***3***	***2***	3	3	40
	Mafiya_takeover	***2***	***4***	4	4	7
	industry	***1***	***1***	***1***	11	11
D4	Mafiya_takeover	***1***	***2***	7	17	19
	industry	***1***	***1***	2	10	11
D5	Mafiya_takeover	***1***	***3***	7	14	35
	industry	***1***	***1***	2	11	11
D6	Mafiya_takeover	***1***	***1***	***1***	***1***	***1***
	industry	***1***	***1***	***1***	10	10



(a)



(b)



(c)

Figure 3.4: Comparing different path lengths for UNICORN

The contribution$_1$ model is chosen for this experiment. The results show that the path length correlates positively with the performance. For *k*=5, the results are close to perfect except for D1 where there is one Mafiya just one spot away from the ideal ranking. The results suggest that it is useful to consider information provided by nodes and links that are multiple steps away from the source node in our framework. Note that the improvement decreases gradually with increasing path length (33% from k=1 to k=2, but only 11% from k=4 to k=5). The deterioration of the improvement shows that the quality of information does not improve linearly with the increasing number of features, which implies a reasonable hypothesis that the farther away a node or link is from the source, the less impact it has on it. The results also suggest a more flexible incremental discovery framework that will be discussed in more detail in Chapter 6.

3.5 Comparing Different Algorithms for Local Node Discovery

Finally, we performed another experiment to evaluate the local node discovery problem described at the end of Chapter 2, where we try to identify nodes that are abnormally connected to a *given source*.

In this experiment, we want to see if UNICORN can identify a crime participant given the other known one as the seed. Our experiment setup is as follows: Since we know that in a gang war there are *two* Mafiya groups involved, we can try to test if we can use one of them to find the other. Similarly, in the industry takeover event there are also two participants, one is a Mafiya node and the other is an industry node that is being taken over. Therefore, we can also test the system by checking if it can

use one to identify the other. In our experiment, we use a gang war participant as the source and ask the system to rank a set of Mafiyas that are abnormally connected to it. We then check whether the other party that is also involved in the gang war is ranked high by the system. We perform a similar test for industry takeover events where we check whether the Mafiya that is abnormally connected to the target industry is the one who is trying to take it over, and vice versa (i.e., whether the industry suspiciously connected to the Mafyia performing industry-takeover is the one that is being taken over).

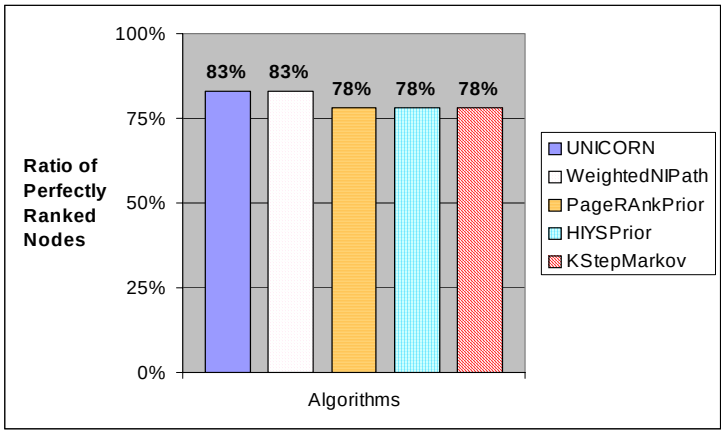
Table 3.5: Finding local abnormal nodes

Data	Results					
	To Be Found	***UNICORN***	WeightedNIPaths	Pagerank Prior	HITS Prior	KStep-Markov
D1	Mafiya_gang1	***1***	16	6	11	7
	Mafiya_gang2	***5***	19	8	9	3
	Mafiya_takeover	***1***	1	1	1	1
	industry	***1***	1	1	1	1
D2	Mafiya_gang1	***1***	1	1	1	1
	Mafiya_gang2	***1***	1	1	1	1
	Mafiya_takeover	***1***	1	1	1	1
	industry	***1***	1	1	1	1
D3	Mafiya_gang1	***1***	1	1	1	1
	Mafiya_gang2	***1***	1	1	1	1
	Mafiya_takeover	***1***	1	1	1	1
	industry	***1***	1	1	1	1
D4	Mafiya_takeover	***2***	9	3	6	4
	industry	***1***	1	1	1	1
D5	Mafiya_takeover	***2***	1	2	2	2
	industry	***1***	1	1	1	1
D6	Mafiya_takeover	***1***	1	1	1	1
	industry	***1***	1	1	1	1

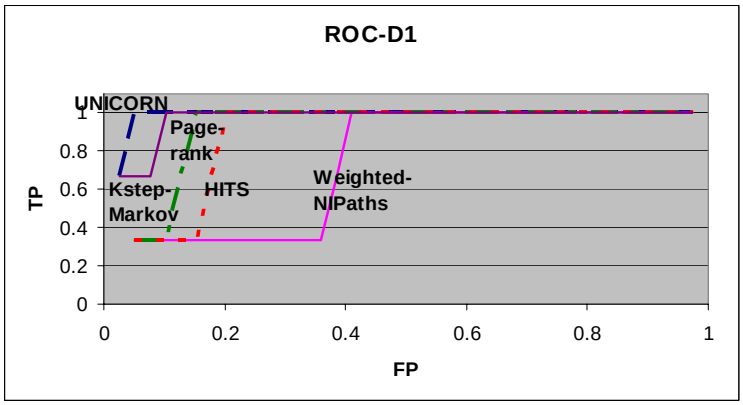
This time we compare UNICORN with the following algorithms: PageRankWithPrior, HITSWithPrior, WeightedNIPaths and KStepMarkov (S. White and P. Smyth 2003). The first two algorithms rank nodes according to their PageRank and HITS value specified for the source node. WeightedNIPath ranks nodes based on the number and length of disjoint paths that lead to a source node (the more shorter paths the better). KStepMarkov ranks nodes based on their stationary probability after performing a *k*-step random walk from the source (the nodes with higher probability will be ranked higher).

The results in table 3.5 demonstrate that for most of the datasets except the hardest ones (i.e., datasets with low observability and high noise), all algorithms can successfully identify the crime participants. One can see that this task seems to be much easier than the previous one, since all algorithms perform significantly better than before. This makes sense, because by revealing a high quality source node we provide a seed suspect, which has a much higher chance to have strong connections to other suspects than an average node. Therefore, finding important or strong connections to a seed suspect is likely to turn up other suspects ("guilt-by-association"), which is a phenomenon also exploited by some group detection algorithms (J. Adibi, H. Chalupsky et al. 2004). The result reveals an interesting phenomenon where most suspects have both important and abnormal connection with other suspects, since they can be detected with both types of mechanisms. However, for the two hardest datasets with low observability and high noise, our algorithm outperforms the others by a decent margin. For the first dataset D1 in the table,

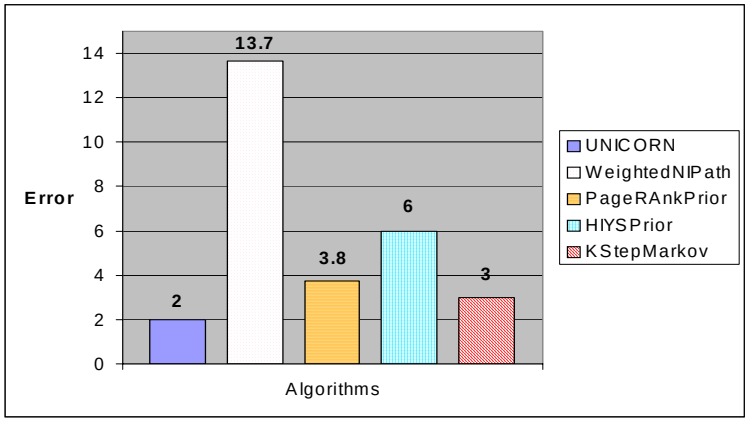
UNICORN ranked the two gang war Mafiyas 1st and 5th while the second best algorithm (KStep-Markov) ranked them 3rd and 7th. The results shown in Table 3.5 do reflect the difficulty of the data. The algorithms perform the worst for D1 and D4, which are the two hardest datasets because of low observability and high noise level. In general our algorithm outperforms others by a larger margin for harder datasets, which implies that using the information provided by different types of relations can be particularly useful for imperfect dataset. Figure 3.5 demonstrates that UNICORN again performs the best for this task based on all three measures.



(a)



(b)



(c)

Figure 3.5: Finding local abnormal nodes

Chapter 4

Explaining Abnormal Instances

In this chapter we describe an explanation-based discovery framework. The basic idea is to augment the discovery system with the capability to generate explanations for its results. To that end, we have designed an explanation mechanism that can summarize the uniqueness of abnormal nodes in a human-understandable form such as natural language. We also describe a human study which demonstrates that using our explanations, users can significantly improve their accuracy and efficiency when trying to identify hidden crime organizers from data.

4.1 Motivation

Explanation plays an important role in many learning systems (J. Pitt 1988; R. Schank and A. Kass 1990; J.S. Dhaliwal and I. Benbasat 1996). Researcher suggests that users would not accept recommendations that emerge from reasoning that they do not understand (S.R. Haynes 2001). We argue that the same argument applies to an anomaly discovery process like ours. For our task, an automatically generated explanation can help verify the discovery system in the sense that the abnormality of nodes are validated by displaying in a human-understand form how they are different from others. Moreover, it also provides users certain information to guide future investigations when necessary. In this sense, the explanation system plays an

important role in the third stage described the discovery process of Figure 1.1 by producing helpful information for further reasoning.

With respect to the application in a homeland security or fraud detection domain, we believe that before marking any individual as an abnormal or suspicious candidate, it is the obligation for a tool or analyst to provide proper justification why such a decision is made to facilitate further assessment of the results. This is particularly important for cases where false positives can cause serious damage to innocent individuals. Automatically generated explanations significantly improve the usability of an information awareness system as well as narrow the gap between an AI tool and its users.

The other reason to develop an automatic explanation generation system is that a KDD system like ours usually deals with very large numbers of individuals and relationships. Given this large data set, it is by no means a trivial job for analysts to manually examine the inputs and outputs in order to come up with explanations such as the ones we are going to provide. In our own experience with the High-Energy Physics Theory (HEP-Th) dataset from the 2003 KDD Cup (S. Lin and H. Chalupsky 2003b), it can take on the order of 30 minutes for a person who is already familiar with the data and features to produce one explanation for one node. Such empirical evidence further strengthens our belief that it is necessary to equip our discovery system with a mechanism that can automatically produce human-understandable explanations.

4.2 Explaining the Uniqueness of Nodes

This section describes a novel mechanism to generate explanations for the abnormal nodes discovered by our system. The underlying concept of our mechanism is that the abnormality of a node can be described by showing how it can be distinguished from the other nodes. The process of generating explanations can be viewed as a kind of summarization that first identifies a small number of key features that cause a node to be unique, and then describes those and their values in a human understandable form.

In general, a semantic profile can contain hundreds or even thousands of path types as features. Our own experience tells us that such an overwhelming amount of information is too much for humans to process. Therefore the first step of explanation generation is to select a subset of features (i.e., path types) that contribute the most to its uniqueness. That is, our system has to determine a small subset of features whose feature values in combination are sufficient to distinguish the abnormal node from the others. In addition, the system has to characterize qualitatively how those selected features in combination can distinguish the abnormal node from the others. Finally, the system has to describe the above process in a human understandable form. In the following, we divide the explanation process into two stages: the first stage focuses on a feature selection and characterization, while the second is a natural language generation stage (see the bottom portion of Figure 2.1).

4.2.1 Feature Selection and Characterization

The problem of feature selection for anomaly explanation is different from the well-studied feature selection problems for clustering. In general, the goal of feature selection is to remove redundant or irrelevant features. However, the definition of redundancy and irrelevancy differs between outlier detection and clustering. Features that are redundant or irrelevant in clustering could carry relevant and important information for anomaly detection. For example, principle component analysis (I. Jolliffe 1986) is a dimension reduction technique that summarizes the most important features for clustering. It throws away the dimensions whose variances are relatively low. Nevertheless, a low variance dimension could carry important information for outlier detection in the sense that the very few instances that are far from the mean in this dimension could indicate abnormality. Consider the case where the majority of authors in a dataset publish one paper per year, with the exception of one author publishes 20 papers per year. In this example the variance in the dimension of "paper publication" could be relatively low, but it is an important dimension to indicate that this particular person is abnormal.

To select a small set of dominant features, we propose to treat explanation as a process of *classification***,** where describing the differences between instances is accomplished by finding a method to classify different types of points into different classes and then characterizing a classification procedure for these classes into a human understandable form. This idea is implemented by assigning a special class label to the node to be explained, and then applying a human-understandable

classifier such as a decision tree to separate the outlier class from the rest. The resulting classification path in the decision tree can then be rendered into an explanation. Note that there is one major difference between our system and a typical classification tool. The goal of a standard classification problem is to develop a system that can *assign the correct label* to the new data, while our explanation system cares more about the *classification procedure.* Therefore classifiers whose classification process is either too complex or opaque (such as, for example, support vector machines or neural networks) are not a good fit for generating human understandable explanations. For this reason, we chose to use a decision tree for our explanation system.

Having modeled the explanation problem as a classification problem, the next question to ask is how many classes we need to explain an outlier and how to define those classes. We propose two different strategies: 2-class and 3-class explanations. In *2-class explanation* the system simply tries to distinguish the abnormal or suspicious points from the rest of the world. To do so we need to divide points into two groups: the outlier class and the normal class and apply a classifier to separate them. In the *3-class explanation* strategy, we introduce an additional class besides the outlier and abnormal class called the reference class, aiming at producing a more meaningful explanation. This is motivated by the observation that while trying to explain something special, one usually prefers using an extra object as reference. For example, the precise interpretation of this explanation “the researcher is abnormal because he is less productive in paper publications” should be “the researcher is

abnormal because he is less productive compared with *other researchers in the same area"*, since it is usually not that meaningful to compare the productivity of a researcher to a journalist, or to the researchers in different fields. In this sense we believe adding an extra reference class (e.g. "other researchers in the same area") might provide a different and sometimes more meaningful view for the uniqueness of a node.

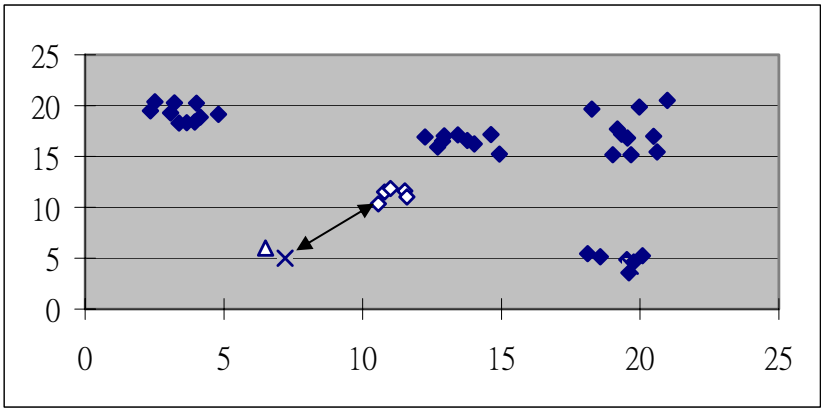


Figure 4.1: 3-class labeling. The "x" point is the outlier to be explained. It together with the triangle point belongs to the outlier class, the white points in the middle belong to the reference class and the rest belong to the global class.

This example conveys a key insight for distance-based outliers: the outlier (i.e., the researcher) is abnormal because it is sufficiently different from a set of people that are to some extent similar to it (i.e., other researchers in the same field). As a result we propose to divide the data points into three classes: the outlier class, the reference class and the normal class. The outlier class contains the outlier point (the "x" in Figure 4.1) and a small number of points that are extremely close to it (e.g., the one indicated by a triangle). The reference class contains points that are *close enough* to the outlier but still possess *sufficient distinction* from the outlier point to allow it to

be abnormal, such as the white diamond points in Figure 4.1. To label the reference points, we first identify the closest gap between the outlier and its neighbors (indicated by the arrow), then label the point (or group of close points) on the other end of this gap as the reference points. The rest of the points are assigned to the normal class. The normal class (e.g., all researchers in the world) is still required in order to provide the global context for the explanation, in the sense that a convincing explanation should not only contain how the outlier class differs from the reference class, but also the position of the reference class in the overall context. Note that it is possible to have multiple layers of reference classes (i.e., the reference of the reference class). The tradeoff is the increasing complexity of explanations.

Once nodes are labeled, we must then describe how they differ in a human-understandable way. To accomplish this, we use a decision tree classifier to generate rules that separate these classes, because decision trees possess the important property of generating easily understandable classification rules (S. Russell and P. Norvig 2003). We designed two different strategies to guide the search in a decision tree. The first is to use the standard information gain heuristic for feature selection, a technique commonly used in decision tree learning. This produces a set of decision rules describing the uniqueness of a node which can then be rendered into explanations such as the following:

uid667 is the only Mafiya that has

larger than 20.8% contribution for [hasMember, ceo]

smaller than 11.1% contribution for [hasMember, sender]

This example generated from the organized crime dataset tells us that uid667 is the only Mafiya that has more than 20.8% chance to be the starting node *s* of a path of type "*s* has some member who is the CEO of a company" and has smaller than 11.1% chance to be the starting node *s* of a path of type "*s* has some member who is the sender of a communication event".

The second strategy is to give higher priority to decision boundary at zero, which we call *zero/non-zero separation*. For the same example node, the system can now produce this explanation:

uid667 is the only Mafiya that has

non-zero [hasMember, ordered_murder]

zero [operates_in_region]

zero [hasMember, victim]

The above decision rules tell us that uid667 is the only Mafiya that has some member who ordered a murder, does not operate in any region, and does not have a member who is a victim in some event.

These two examples illustrate the pros and cons of both strategies. For the regular decision tree strategy, it is possible to precisely and concisely separate the outlier from the other nodes with a shallower tree. However, to make sense out of the explanation, users have to think in terms of relative probability or contributions (e.g. larger than 20%), which is somewhat less natural for users to understand. The process becomes even harder when the users must perform reasoning in terms of MI or PMI values such as *"has higher than 12.4 mutual information value …"*. Therefore we

believe that the *contribution measure* is the most favorable model for explanation purposes, as it is easier to digest and still generates high quality results. For the zero/non-zero separation strategy, the system provides the user a simplified explanation without having to resort to an exact feature value. We believe that it is more understandable in view of the fact that the users only have to reason about path type features that *involve* or *do not involve* the node. The drawback of this strategy is that it conveys less information, is less effective in separating points and generally requires more rules to distinguish abnormal nodes from the rest.

It is reasonable to infer that the first strategy is more suited for dense datasets, which require more detailed information to make the distinction and where the benefit of fewer rules outweighs the increase of complexity caused by having to reasoning with probabilities. Conversely, the second strategy is more suitable for sparse datasets in which the zero/non-zero separation is sufficient and can provide a more intuitive explanation. The experiments described in Chapter 5 will shed further light on these trade-offs.

An important feature of our explanation framework is that it is completely node independent. Given a set of semantic profiles, it can explain abnormal points as well as normal points. To explain the not-so-abnormal points generally requires more decision rules, resulting in longer explanations. In fact, the minimum description length criterion can be utilized as a novel outlier ranking criterion, which ranks the abnormality of points based on their explanation description length. We will elaborate upon idea in Chapter 6.

4.2.2 Generating Natural Language Explanations for Suspicious Nodes

The previous section describes how to select a subset of features and their decision boundaries. The next step is to translate those results into a natural language such as English. Note that up to the feature selection stage, the process is completely unsupervised and domain independent, which means we need neither domain knowledge nor training examples to create certain level or explanations, such as the *uid667* examples illustrated in the previous section. However, to translate our results into natural language, it is necessary to have certain language-dependent domain knowledge (e.g. the link "*ordeContractMurder*" means "*somebody ordered a contract murder*" in English), or some training examples that system can learn from.

To produce natural language outputs, we focus on two issues. The first is to translate a path or path type in the semantic graph into its corresponding English. For example, we want to translate the following path $S \xrightarrow{emails} ? P \xrightarrow{father\_of} ? Q \xrightarrow{travels\_to} T$ into "S sends an email to somebody whose child travels to T". We designed an n-gram based supervised learning system together with a small set of rules to translate paths into English sentences. Since this is not the major focus and contribution for this thesis, we point the interested reader to the technical details contained in (S. Lin 2006). Therein, experimentation shows that our system can translate 82% of the paths accurately and fluently (S. Lin 2006).

Once we can produce the corresponding natural language description for each feature, the second step is to combine all the information to create meaningful explanations. We do this via four different natural language generation templates for

2-class and 3-class explanations and for regular and zero/non-zero decision boundaries. Details of the templates are given in Appendix II.

Here are some example explanations for abnormal nodes generated by our system from three different types of datasets (organized crime, movie and bibliography). More results will be shown in Chapter 5.

2-class explanation, zero/non-zero separation is on, from the movie dataset
Salvador Dali is the only 1 actor in the dataset (which contains a total of 10917 candidates) that *is the visual director of some movie , and* *is the writer of some movie, and* *never is the sibling of some movie person*
3-class explanation, zero/non-zero separation is on, from the organized crime dataset
uid3491 is one of the only 10 Mafiyas in the dataset (which contains a total of 42 candidates) that *–never has some member receiving some money from somebody, and* *–never hired some killer, and* *–never has some member who is a perpetrator* *However, uid3491 is different from the other 9 Mafiyas because it* *–has some middleman participating in some meeting, while others don't* *–has a member who ordered a contract murder, while others don't*
2-class explanation, zero/non-zero separation is off, from the bibliography dataset
h_lu is the only 1 author in the dataset (which contains a total of 7152 candidates) that *–has higher than 1.71% chance to be the starting node S of the paths representing* *"S wrote paper with somebody who wrote paper with another person"*

The first example is the most simplified one and can easily be understood. It utilizes 2-class explanation with zero/non-zero separation to explain what makes Salvador Dali different from the others actors. It basically says Dali is an interesting actor because, unlike other actors, he is also a visual director, a writer, and doesn't have any sibling working in the movie industry.

The second explanation uses 3-class explanation, which first classifies the node into a category that contains small amount similar nodes, and then pinpoints the difference between the node and the others in this category. In the last example, the zero/non-zero separation is set to off—hence the system uses the contribution values to separate nodes.

4.3 Explanation Algorithm

This section describes some implementation details and the pseudo code (*UNICORN_EXP*) for the explanation system.

```
function UNICORN_EXP (Sp[][], n, 2_class, zero_sep, exp_num)
1.  array Lab[] := assign_label (Sp, n, 2_class);
2.  if (2_class)
3.   rules:=feature_sel (Sp, Lab, n, zero_sep,"outliers","normal", exp_num);
4.  else{
5.     rules:= feature_sel(Sp_sub, Lab, n, zero_sep, "outliers", "reference", exp_num);
6.    forall k s.t. (Lab[k]="outlier" or Lab[k]="reference")
7.       Lab[k]:="outlier_reference";
8.     rules:+= feature_sel (Sp_sub, Lab, n, zero_sep,"outlier_reference", "normal",
        exp_num );
9.  return NLG(rules)
```

UNICORN_EXP takes five inputs as follows:

1. *Sp* points to the semantic profiles of all the nodes in the network. For example, *Sp[k]* returns the semantic profile of node *k*.
2. *n* points to the source node to be explained.
3. The Boolean variable *2_class* is true if two-class explanations are used and false for three-class explanations.

4. The Boolean variable *zero_sep* is true if applying the zero/non-zero separation heuristic.
5. Integer *exp_num* constraints the maximum amount of features the explanation system can use to generate the explanations.

The function *assign_label* returns the class label of each point (see below for details). The *feature_sel* function returns a set of rules that separate one given class from another (see below for details). Note that in the three-class explanation, we execute *feature_sel* twice, the first time to isolate the "outlier" class from the "reference" class and the second time to separate the "outlier and reference" class from the "normal" class. Finally the NLG function translates the rules into natural language.

Function *assign_label* takes a semantic profile, a point to be explained, and a Boolean variable (*2_class*) as inputs and outputs an array which represents the class labels of every node. For 2-class explanation, the system simply marks the outlier and its close neighbors as "outlier" and the rest as "normal". This *near(point1, point2)* function returns true if the distance between point1 and point 2 is below a given threshold. For 3-class explanation, it first identifies the gap point through the *get_gap_point function.* The gap point (e.g. the leftmost white diamond point in Figure 4.1) represents the boundary between the outlier class and reference class. All the points whose distance to the outlier point is smaller than that of the gap point are marked as outliers. Those that are very close to the gap point are marked as the "reference" class and the rest are marked as the "normal" class.

The function *feature_sel* basically uses a decision tree generator to produce the decision rules to separate the classes. If zero/non-zero separation is used, then it modifies all the features to binary ones before executing the decision tree. The tree returns a set of rules (i.e., a decision path in the tree) to separate two given class *s1* and *s2*. *sub_path* function truncates the decision paths if it is longer than a given number *exp_num*.

```
Function assign_label(Sp, n, 2_class)
 1.   array Lab[];
 2.   if (2_clas )
 3.    for k:= 0 to Sp.size-1
 4.      if (near(Sp[k],Sp[n]))
 5.       Lab[k]:=”outlier”;
 6.      else
 7.       Lab[k]:=”normal”;
 8.   else
 9.     g=get_gap_point(Sp,n);
 10.    for k:= 0 to Sp.size-1
 11.     if (near(Sp[k], Sp[g]))
 12.       Lab[k]:=”reference”;
 13.     else if (distance(Sp[k],Sp[g])<distance(Sp[n],Sp[g]))
 14.       Lab[k]:=”outlier”;
 15.     else Lab[k]:=”normal”
 16.  return Lab;
```

function *get_gap_point (Sp,n)*

1. ***array*** *L[]= sort_distance (Sp,n);*
2. ***for*** *m:= 1* ***to*** *k //k is the maximum number of neighbors allowed for an outlier*
3. ***array*** *gap[m-1]:=distance(Sp[n],Sp[L[m]])-distance(Sp[n],Sp[L[m-1]]);*
4. ***return*** *argmax gap(x) // it returns the number x that maximizes gap(x)*

function *feature_sel (Sp, Lab, n, zero_sep, s1 , s2, exp_num)*

1. ***if*** *(zer0_sep)*
2. *Sp:=to_binary(Sp); //modify the feature values into binary values*
3. ***path*** *DT:= decision_tree (Sp, Lab, s1, s2)*
4. ***return*** *sub_path(DT, exp_num)*

4.4 Evaluating the Explanation System

In this section, we describe an evaluation for the usefulness of our explanation system based on the synthetic organized crime dataset. The goal is twofold: first we want to know whether the explanations generated by our system can assist human subjects to make more accurate and confident decisions in terms of identifying the hidden crime organizers. Second, we want to know whether the explanations can reduce the time needed to make these identifications. To answer these questions we designed three tasks. In Task 1 we provide the subjects the original data as a file (containing 5800 nodes and 26000 links, sorted by nodes) with English translations for each relation as the control set. We then ask them to select three Mafiyas from ten given candidates that they believe are most likely to have participated in the gang wars (two) and industry takeover (one) by any means. In Tasks 2 and 3, we provide 2-class and 3-class zero/non-zero explanations for the same ten candidates and ask

the subjects to perform the same task based only on the provided explanations. We record the time (we limit the maximum time allowed to 60 minutes) and confidence (from 0 to 4, 0 means no confidence at all) for each task. To avoid interference and bias among different tasks, we not only modified the names of the candidates between tasks, but also told the users that they are from different datasets. We tested on 10 human subjects. The results show that while working on only the original network, no subject can identify all three candidates correctly, and only two subjects find at least one target Mafiyas. With only 2-class explanations, 80% of the subjects are able to identify all three candidates. With only 3-class explanations, 20% of the subjects can identify all three candidates, while 60% of the subjects found two out of the three. Table 4.1 shows the subjects' performance for each individual task.

Table 4.1: Evaluation results for explanation system. In Task 1 the subjects are given the original dataset. In Task 2 and 3 the subjects are given the 2-class and 3-class explanations, respectively. The numbers in the first three columns stand for the percentage of subjects who successfully identified the crime participants for each task.

	Mafiya_takeover	Mafiya_gang1	Mafiya_gang2	Avg confidence	Avg time
Task 1	10%	10%	10%	0.3	60min
Task 2	90%	90%	90%	2.2	22.5min
Task 3	70%	60%	60%	1.95	23min

In Task 1, for each Mafiya group sought only one human subject did successfully identify it within the time limit. 6 of the 10 subjects gave up after spending less than 60 minutes on it for the reason that they believed the data to be too

complicated for them to analyze. This is understandable given the thousands of nodes and links where subjects have to keep track of 814 different path types (up to length 4) like the system does. Our human subjects' feedbacks indicate that reasoning for indirectly connected links and nodes is very hard for humans, in particular compared with the machine which only took seconds to generate the semantic profiles for the Mafiyas and produce the explanations. Note that the times recorded here are the average time for the subjects who reported at least one correct candidate. The confidence level for Task 1 is close to 0 (0.3). The results demonstrate that the original network with baseline explanation (i.e., simply translating every single relation into English) is very difficult for humans to analyze within a limited amount of time. The results in Task 2 show that with the 2-class explanation, the human subjects improve dramatically (90%) in identifying the suspicious candidates with much less time (22.5 min) spent and much higher confidence (2.2). For 3-class explanations, the accuracy is a little bit lower (60%-70%), but the confidence (1.95) and time required (23min) are similar to Task 2. According to the feedback from subjects, most of them thought the 2-class explanation strategy provided sufficient amount of information for them to perform the analysis while the information provided by 3-class explanations seemed to be a bit overwhelming. We think this is the major reason why they generally performed not as well for Task 3 compared with Task 2. Appendix III lists the 2-class explanations and 3-class explanations generated by our system for this study.

The results of this human study demonstrate that with the explanations generated by our system, users have a significantly higher chance to identify the truly suspicious instances within less amount of time.

Chapter 5

Applying UNICORN to Natural Datasets

In this section we demonstrate how UNICORN can find abnormal nodes in several real-world natural datasets. Our goal is to demonstrate how in general the complete system works as well as to show that the system is domain independent and can be applied to find abnormal (and sometimes interesting) instances in arbitrary semantic graphs. We will also demonstrate that different explanation strategies are suitable for different types of datasets. For each dataset, we will list a set of abnormal instances our system finds and their explanations. To make the explanations more understandable, we chose to use at most three features in explanations based on contribution_1 analysis, and limit the maximum path length to 4.

The first network is generated from the KDD Movie dataset designed by Gio Wiederhold and available from the UCI Machine Learning Repository. The second network is generated from the HEP-Th (High Energy Physics Theory) bibliography dataset used in the 2003 KDD Cup. We provide most examples from the movie dataset because it is a domain that is familiar to most people..

5.1 KDD Movie Network

From the KDD Movie dataset, we extracted 37808 nodes representing movies (11540), directors (3233), producers (2867), actors (18777), and some other movie-

related persons (5772). There are people who play multiple roles in this dataset (e.g. both director and actor). We also extracted about 150,000 inter-node relations. We must note that because the data was created manually, errors or missing values do exist and they are inevitably carried into our network. There are 42 different relation types in this dataset (21 basic and 21 inverse relations), which can be divided into three groups: relations between people (e.g. spouse, mentor, worked with), relations between movies (e.g. remake), and relations between a person and a movie (e.g. director, actor). Below are the descriptions of the 21 basic link types (the numbers at the end of each line represent their occurrence statistics).

actor (<person>, <movie>): <person> acted in <movie>(46529)
affected_by (<person1>,<person2>): <person1> was affected by <person2>(56)
authors (<person>, <movie>): <person> wrote a book adapted into the <movie>(1829)
c (<person>, <movie>): <person> is the cinematographer of <movie>(986)
child_of (<person1>,<person2>): <person1> is the child of <person2>(208)
d (<person>, <movie>): <person> directed the <movie>(11541)
e (<person>, <movie>): <person> is the editor of the <movie>(51)
g (<person>, <movie>): <person> is the choreographer of the <movie>(45)
lived_with (<person1>,<person2>): <person1> lived with <person2>(393)
lovername (<person1>,<person2>): <person1> is the lover of <person2>(16)
m (<person>, <movie>): <person> composed music for <movie>(1311)
member_of (<person1>,<person2>): <person1> is the member of <person2>(58)
mentor_of (<person1>,<person2>): <person1> is the mentor of <person2>(32)
p (<person>, <movie>): <person> produced the <movie>(5806)
parent_of (<person1>,<person2>): <person1> is the parent of <person2>(97)
remake (<movie1>, <movie2>): <movie1> was remade from <movie2>(1304)
sibling_of (<person1>,<person2>): <person1> is a sibling of <person2>(306)
spousename (<person1>,<person2>): <person1> is the spouse of the <person2>(1372)
v (<person>, <movie>): <person> is the visual director of the <movie>(254)
w (<person>, <movie>): <person> wrote the script for <movie>(3491)
workedwith (<person1>,<person2>): <person2> worked with <person2>(1112)

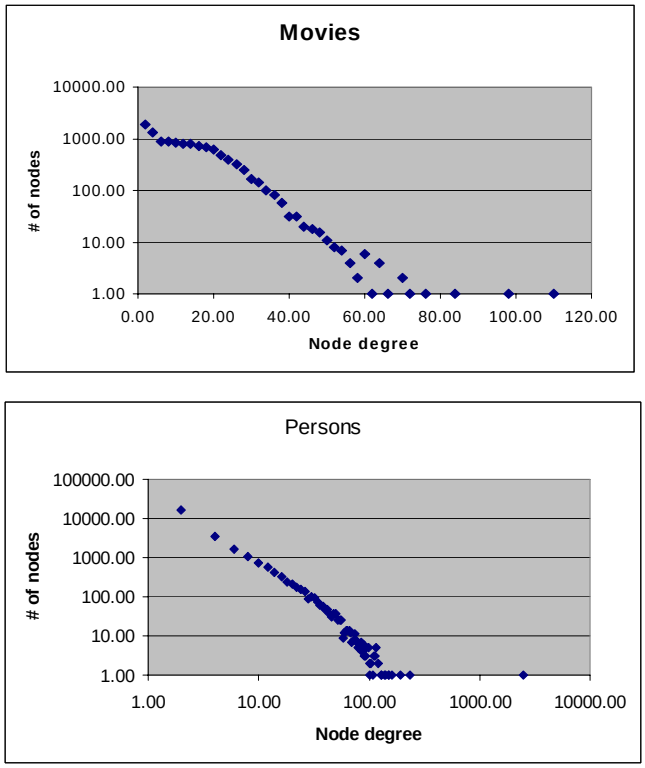


Figure 5.1: Degree histogram for movies and persons in KDDMovie network

Figure 5.1 displays the degree-histogram of the people and movies. The *x*-axis represents the degree of the nodes and the *y*-axis represents the number of nodes. One can see that the majority of movies and persons are of low degree values, but there are still a few of that have many connections with others. The person degree roughly satisfies the Zipf's law. The movie of the highest degree is *Around the World in 80 Days*. The person of the highest degree is Hitchcock.

5.2 Explaining Why Hitchcock is Abnormal

In our experiment, we chose the relation-only constraint and limit the maximum path length to four. Table 5.1 displays four different explanations for uniqueness of

Hitchcock, which our system believes to be the person with the most abnormal semantic profiles in this dataset based on Ramaswamy's distance-based outlier algorithm. This makes sense, because the dataset creator described himself as a fan of Hitchcock and the dataset has a significant Hitchcock bias, which makes him very different from all other nodes.

Table 5.1: Four types of explanations for Hitchcock in the KDD Movie network

2-class explanation, zero/non-zero separation is on
Hitchcock is the only 1 actor in the dataset (which contains a total of 10917 candidates) that *is the mentor of some movie person, and* *is affected by some movie person*
2-class explanation, zero/non-zero separation is off
Hitchcock is the only 1 actor in the dataset (which contains a total of 10917 candidates) that *has larger than 4.148% chance to be the starting node S of paths of type "S directed some movie"*
3-class explanation, zero/non-zero separation is on
Hitchcock is one of the only 2 actors in the dataset (which contains a total of 10917 candidates) that *is affected by somebody who wrote a movie that has some producer* *Moreover, Hitchcock is different from the other 1 actor because it* *Acted in a remade movie that has some director, while others didn't*
3-class explanation, zero/non-zero separation is off
Hitchcock is one of the only 2 actors in the dataset (which contains a total of 10917 candidates) that *has larger than 43.243% chance to be the starting node S of paths of type* *"S is affected by some actor"* *Moreover, Hitchcock is different from the other 1 node(s) because it* *has much lower (20% v.s 80%) chance to be the starting node S of paths of type* *"S is affected by somebody who wrote a movie that has some producer"*

In the first explanation our system extracts two features (i.e., the non-zero contribution of two path types) to separate him from the rest of the world. It basically

says that the reason Hitchcock is abnormal is that he is not only the mentor of somebody but also affected by others in the dataset. This might be interesting to some movie buffs to further investigate who those persons are and why the others in the dataset do not have this characteristic. In this dataset both "mentor_of" and "affected_by" relations are not very common, and most of them are related to Hitchcock, which makes him abnormal. The next explanation conveys key information about Hitchcock's uniqueness by showing that he *directed the most movies.* In the third explanation, the system first isolates Hitchcock together with a set of (in this case, one) similar actors and differentiates them from the rest of the world, and then uses another classifier to explain the difference between Hitchcock and his closest neighbor (remember that the system applies the classifier twice for 3-class explanation). The last explanation uses a similar strategy as the third, but this time it does not limit itself to zero/non-zero decisions. It conveys the idea that *"among the actors that are affected by some other actors frequently, Hitchcock has a lower chance to be affected by a person who is both a writer and a producer"*, which makes him unique. All these explanations seem to be meaningful and to some extent interesting in our opinion, which implies three things: First, using path types as features seems to be a good choice for explanation purposes, since the linear combination of relations can be translated into meaningful English, and facilitates the generation of easily understandable explanations. Second, using the contribution dependence measure (or its simplified zero/non-zero version) can produce intuitive explanations. Third, trying to explain the *uniqueness* of a node seems to be a good

strategy to find out potentially interesting information about it. Note that although the four explanations are different, they are to some extent correlated (except the second one) in the sense that some similar features (i.e., those related to “mentor_of” and “affected_by”) are selected to generate explanations.

5.3 Finding and Explaining Abnormal Movies, Directors, and Actors

Tables 5.2 to 5.4 present some examples of abnormal movies, directors, and actors generated from the KDD Movie dataset. In these examples we use the explanation mechanism itself to determine who is abnormal. To do that, UNICORN first generates the semantic profile for each node, and then applies the explanation mechanism to explain them. Based on its own explanation, we can assign each node into one of two groups, the *abnormal* group and the *not-so-abnormal* group. A node belongs to the *abnormal* group if UNICORN can explain its uniqueness by three or fewer features using 2-class explanation while zero/non-zero separation is on; otherwise, it was assigned to the not-so-abnormal group. We then manually picked some nodes from the abnormal group which we believe to be better known by people for display. It can be easily seen that the zero/non-zero criteria generally produces explanations that are simpler and easier to understand at the expense of skipping some detail information. Note that the indented parts of the explanations are the translations of path types generated by our path-to-sentence generator. For example, in the first explanation of Table 5.2, the sentence “has a composer who is also an actor” comes from the path *[has_composer, acted_in]*.

Table 5.2: Examples of abnormal movies from the KDD Movie dataset

2-class explanation, zero/non-zero separation is on
"Snow White and the Seven Dwarfs" is the only 1 movie in the dataset (which contains a total of 9097 candidates) that *has a composer who is also an actor, and* *is remade into some movie adapted from a book, and* *is remade into some movie that has a composer*
2-class explanation, zero/non-zero separation is off
"Phantom of The Paradise(1974)" is the only 1 movie in the dataset (which contains a total of 9097 candidates) that *has larger than 5.31% chance to be the starting node S of paths of type "S has been remade from a movie that has some visual director"*
3-class explanation, zero/non-zero separation is on
"Superman" is one of the only 24 movies in the dataset (which contains a total of 9097 candidates) that *has a composer who worked with some movie person, and* *never has a writer, and* *has some actor whose spouse also works in the movie industry* *Moreover, "Superman" is different from the other 23 movies because it* *has a visual director who is also a composer, while others didn't*
3-class explanation, zero/non-zero separation is off
"Romeo and Juliet" is one of the only 4 movies in the dataset (which contains a total of 9097 candidates) that *has larger than 1.974% chance to be the starting node S of paths of type "S is remade into some movie that has some cinematographer"* *Moreover, "Romeo and Juliet" is different from the other 3 node(s) because it* *has much higher (66.667% v.s 0%) chance to be the starting node S of paths of type "S is remade into some movie whose producer is also an editor"*

Remember that most of the movie data in this dataset was collected manually and is to some extent biased by the interests of the dataset author. Therefore, the data is somewhat skewed and also has missing values for certain movies. For example, in the third explanation of Table 5.2 it says the movie *Superman* "never has a writer", while the real-world interpretation is that the person who wrote the script for *Superman* is not recorded in the dataset. This example also demonstrates that UNICORN can potentially be applied as a data cleaning tool.

Table 5.3 lists some abnormal directors. In the first explanation it says Woody Allen never acted in some movie that has producer. This is again due to the producer information for the movies he acted not being recorded. The second explanation tells us that what makes Stephen King unique is that he has the highest chance to be a director who wrote a book adapted for a movie, which is consistent with the general impression that King is better known as an author than as a director (therefore he is abnormal as a director). The last explanation is also interesting. It tells us that among the directors that mentored others more frequently, Yamamoto is different from them, because he also affected people who already have some other mentors.

Table 5.4 explains some abnormal actors UNICORN found, and shows some interesting facts about these people. For example, Elton John's co-composer has authored some book that was adapted for some movie.

Table 5.3: Examples of abnormal directors from the KDD Movie dataset

2-class explanation, zero/non-zero separation is on
Woody Allen is the only 1 director in the dataset (which contains a total of 3233 candidates) that *acted in some movie that was remade into some other movie, and* *never acted in some movie that has a producer*
2-class explanation, zero/non-zero separation is off
Stephen King is the only 1 director in the dataset (which contains a total of 3233 candidates) that *has larger than 36.508% chance to be the starting node S of paths of type "S wrote some book adapted for a movie"*
3-class explanation, zero/non-zero separation is on
George Lucas is one of the only 61 directors in the dataset (which contains a total of 3233 candidates) that *directed some movie that has some composer, and* *never directed some movie adapted from some book, and* *directed some movie that was remade into some movie* *Moreover George Lucas is different from the other 60 directors because it* *is the writer of some movie that has some visual director, while others aren't*
3-class explanation, zero/non-zero separation is off
Yamamoto is one of the only 8 directors in the dataset (which contains a total of 3233 candidates)that *has larger than 2.419% chance to be the starting node S of paths of type "S is the mentor of some director"* *Moreover, Yamamoto is different from the other 7 node(s) because it* *has much higher (50% v.s 0%) chance to be the starting node S of paths of type "S affects somebody who is mentored by some other person"*

Table 5.4: Examples of abnormal actors from the KDD Movie dataset

2-class explanation, zero/non-zero separation is on
Antonio Banderas is the only 1 actor in the dataset (which contains a total of 10917 candidates) that *lived with some movie person who is the child of some movie person, and* *lived with some movie person whose spouse is also a movie person, and* *is the spouse of some actor*
2-class explanation, zero/non-zero separation is off
Elton John is the only 1 actor in the dataset (which contains a total of 10917 candidates) that has 100% chance to be the starting node S of paths with path type "S is the composer for some movie which has another composer who wrote a book adapted for a movie"
3-class explanation, zero/non-zero separation is on
Fassbinder is one of the only 54 actors in the dataset (which contains a total of 10917 candidates) that *wrote a book adapted for a movie, and* *never is the composer of some movie, and* *never is the child of some movie person* *Moreover, Fassbinder is different from the other 53 actors because it* *cinematographed some movie, while others didn't* *mentored by some movie person, while others didn't*
3-class explanation, zero/non-zero separation is off
Brad Pitt is one of the only 2 actors in the dataset (which contains a total of 10917 candidates) that has larger than 28.571% chance to be the starting node S of paths of type"S lived with some movie person who is the child of some director" *Moreover, Brad Pitt is different from the other 1 node(s) because it has much lower (0% v.s 50%) chance to be the starting node S of paths of type "S wrote some movie which has another writer who lived with a movie person"*

The results from Table 5.1 to Table 5.4 demonstrate that UNICORN can identify abnormal instances, and the explanations it produces are actually sometimes interesting. We observed that it can identify at least three types of interesting things:

1. Some facts users might not have known before. For example, that George Lucas has never directed some movie adapted from some book and that he is a writer for some movies.
2. Something that matches users' original impressions about an instance. For example, the system finds that "authoring books" makes Stephen King unique as does "composing" for Elton John, which matches our general impression of them (i.e., that King is a famous writer and that John is a well-known musician). This is interesting considering our system does not have any background knowledge about nodes.
3. Strange things that are potential errors. For example, Woody Allen never acted in some movie that has a producer.

Note that given the semantic profiles our explanation system can explain not only abnormal nodes but also not-so-abnormal nodes. For example:

*L. Anderson is one of the only **48** actors in the dataset (which contains a total of 10917 candidates) that*

produced some movie

directed some movie

never is the writer of some movie

In this case, the best job our 2-class explanation system can do based on three zero/non-zero features is to separate L. Anderson from the other (10917-48=10869) nodes. In other words there are still a group of 47 actors that are similar to Anderson. This example suggests that our explanation system can be used to generate explanations for every node regardless of whether it is unique or not. Moreover, the *structure* and *size* of explanations reveals the degree of abnormality of the nodes. This motivates another idea of using our explanation mechanism for outlier ranking, which will be discussed in more detail in Chapter 6.

5.4 HEP-Th Bibliography Network

The second natural dataset we used is the high-energy physics theory (HEP-Th) bibliography dataset, which records papers, their authors and citations in this domain. For the HEP-Th dataset, we extracted six different types of nodes and twelve types of links to generate the MRN. Nodes represent papers (29014), authors (12755), journals (267), organizations (963), keywords (40) and the publication time encoded as year/season pairs (60). Numbers in parentheses indicate the number of different entities for each type in the dataset. There are about 43000 nodes and 477,000 links overall. We defined the following types of links to connect nodes:

writes (<author>,<paper>): *<author> writes <paper> (57447)*
date_published(<paper>,<time>): *<paper> published at <time> (29014)*
organization(<author>,<organization>): *<author> belongs to <organization> (11060)*
published_in(<paper>,<journal>): *<paper> is published in <journal> (20715)*
cites(<paper1>, <paper2>): *<paper1> cites paper <paper2> (342437)*
keyword_of(<paper>,<keyword>): *<paper> has <keyword> (16714)*

These links are viewed to be directional with implicit inverse links. Thus, there are a total of 12 different relation types. Figure 5.2 describes the degree histogram for this dataset.

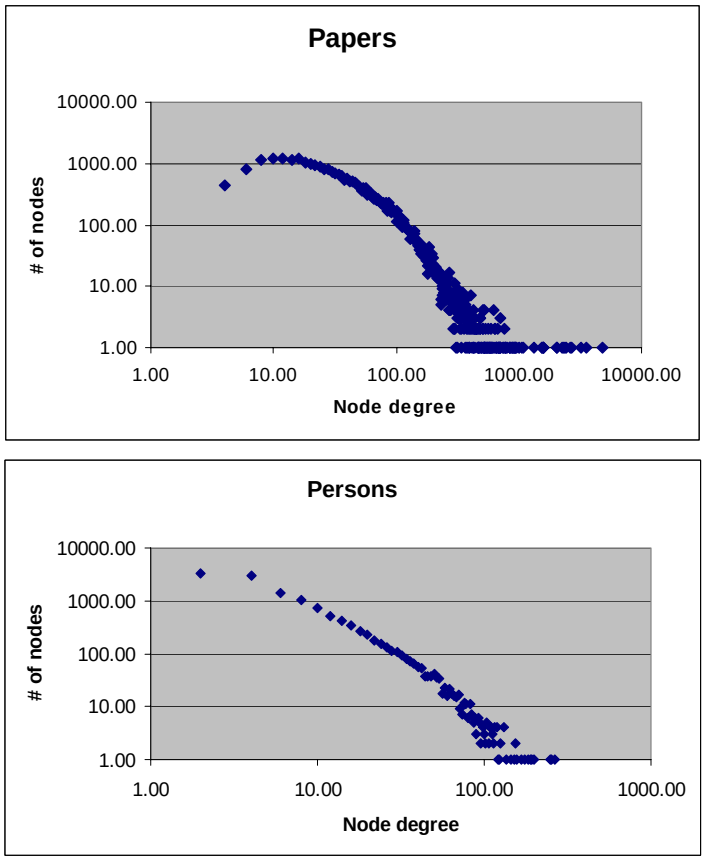


Figure 5.2: Degree histogram for papers and persons in the HEP-Th network.

Table 5.5 displays some results about abnormal authors in the HEP-Th dataset with zero/non-zero separation turned off. Here we selected the top candidates ranked by Ramaswamy's distance-based outlier algorithm. We exploit the loop Meta-constraint together with contribution$_1$ feature value.

In the 2-class explanation, we can see that C.N. Pope is abnormal because he has a higher chance than anybody else to cite his own papers. From the 3-class explanation, we can conclude that among the persons whose papers have higher chance to be cited together by another paper, and who have a higher chance to publish multiple papers in the same journal, Edward Witten is the one who has the highest chance to have papers being cited together by another paper. It turns out Witten is a very influential person in HEP-Th (two of the top three cited papers in this dataset are authored by him). On the other hand, Table 5.6 shows that using three zero/non-zero features is not sufficient to describe the uniqueness of these people, since there are still respectively 1144 and 1540 other candidates that satisfy the same conditions.

Table 5.5: Abnormal researchers in the High-Energy Physics dataset and their explanations with zero/non-zero separation turned off.

2-class explanation, zero/non-zero separation is off
C. N. Pope is the only 1 node in the dataset (which contains a total of 7152 candidates) that *has higher than 1.01% chance to be the starting node S of paths of type "S wrote a paper that cites his/her own paper"*
3-class explanation, zero/non-zero separation is off
Edward_Witten is one of the only 2 nodes in the dataset (which contains a total of 7152 candidates) that *has higher than 1.71% chance to be the starting node S of paths of type "S has multiple papers cited by one single paper", and* *has higher than 0.615% chance to be the starting node S of paths of type "S has multiple papers published in one journal"* *Moreover, Edward_Witten is different from the other 1 node(s) because it* *has slightly higher (4.04% v.s 1.71%) chance to be the starting node S of paths of type " S has multiple papers cited by one single paper".*

Table 5.6: Abnormal researchers in the High-Energy Physics dataset and their explanations with zero/non-zero separation turned on.

2-class explanation, zero/non-zero separation is on
C.N. Pope is one of the only 1145 nodes in the dataset (which contains a total of 7152 candidates) that *published two or more papers with the same keyword, and* *never has a colleague that once belonged to the two institutes he has ever belonged to, and* *has coauthor came from the same organization*
3-class explanation, zero/non-zero separation is on
Edward_Witten is one of the only 1541 nodes in the dataset (which contains a total of 7152 candidates) that *published two or more papers with the same keyword, and* *published two or more papers at the same season, and* *has coauthor came from the same organization*

Comparing the HEP-Th with the KDD Movie network, the former has more nodes and links but much fewer relation types (only 12 compared with 44 in KDD Movie). This fact makes the HEP-Th dataset a much denser dataset with more points and fewer features compared with the movie dataset. In a dense dataset, the zero/non-zero separation strategy is not as useful, because we need more precise information to classify a node as demonstrated in Table 5.6.

5.5 Explaining Local Abnormal Nodes

The explanation system can be easily adapted to explain abnormal local nodes (i.e., nodes abnormally connected to a given source node). For example, the following is the 3-class zero/non-zero explanation our system generates to explain an organization that is abnormally connected to Dr. Chen from the HEP-Th dataset:

NCU is one of the only 3 organizations in the dataset (which contains a total of 187 candidates)that

C.M. Chen once belonged to

Moreover, NCU is different from the other 2 nodes because it

never is the organization of some person whose paper is cited by a paper written by C.M. Chen,

never is the organization of some person whose paper cites paper written by C.M. Chen,

This example shows that NCU is one of the three organizations Dr. Chen has ever belonged to. Furthermore, it is “abnormal in the abnormal” because unlike the other two, Dr. Chen does not have any citationship or collaboration with the people from NCU. This result is interesting in the sense that the system can identify that the organizations he belongs to are an important feature to characterize him (i.e., “Type 2” results). Moreover, it also figures out that what makes him even more abnormal is that he does not have any other connection with one of the organization he once belonged to.

Chapter 6

Discussion and Applications

The previous chapters described a series of methods designed to handle the anomaly detection problem in an MRN. A common feature of the discussed techniques is that they are designed to be general and domain independent, will therefore, be potentially applicable to a variety of problems. This chapter discusses the scalability issues of our system and proposes a number of applications that can be solved by the techniques we have developed.

6.1 Scalability and Efficiency of UNICORN

In this section, we go through each stage of UNICORN to discuss scalability and efficiency issues as well as their solutions.

6.1.1 Feature Selection

In the feature selection stage, UNICORN applies a set of meta-constraints to generate a set of path types as features. The worst-case complexity for the default relation-only meta-constraint is exponential: $O(\frac{r!}{(r-k)!})$, where r is the number of different relation types and k is the maximum path length. This is because, in the worst case, each permutation of relation types becomes a possible feature. Complexity, therefore, a huge number if both r and k are large. Fortunately, as described in the previous chapters, it is intuitively clear that the farther a node/link is

away from the source node, the less impact it has on the source. Therefore, in general a small *k* is sufficient, and our experiment results support this hypothesis. Also note that not all permutations of relations are meaningful and exist in the network. Furthermore, the result of Figure 3.4 demonstrates that the improvement rate of the discovery quality tends to degrade with the increase of path lengths. Based on this observation, we designed an incremental version for UNICORN, called UNICORN_Inc shown below (without loss of generality, we assume that UNICORN utilizes the outlier detection algorithm, which returns a set of abnormal nodes instead of a ranking):

function *UNICORN_Inc (G, mc, k)*

1. ***set*** *L=UNICORN(G, mc, 1);*

2. ***for*** *x:= 2* ***to*** *k*

3. ***set*** *L'= UNICORN(G, mc, x)*

4. ***if*** $\frac{|L \cap L'|}{|L \cup L'|} > Threshold$ *,* ***then return*** *L';*

5. ***else*** *L=L';*

6. ***return*** *L';*

Starting from path length 1, UNICORN_Inc executes UNICORN incrementally in terms of maximum path length *k*. At the end of every iteration, it checks how the results are changed compared with the previous iteration by comparing the overlap between the current and previous results. If it is larger than a pre-determined threshold (which implies that the outputs are very similar), then the system stops and

returns the current output, otherwise it will keep increasing the path length. Another advantage of UNICORN_Inc is that it is an *anytime algorithm*, which means the user can stop it at any time but still be able to get results of a certain quality.

6.1.2 Feature Value Generation

Feature value generation is generally the most computationally costly stage of UNICORN. Recall that in this stage, UNICORN has to count paths in the MRN in order to compute the statistical dependencies between nodes and paths. The worst-case time complexity for this stage is $O(|Path|)$, which means that it has to go through all paths up to certain length in the network. Although this number highly depends on the network topology, it is usually very large in the sense that the number of paths tends to increase exponentially with the average branching factor of the network.

Fortunately, our design of random experiments provides a natural way for us to approximate the feature values. Since feature values are based on the dependency of two random variables of a particular *path-selection* random experiment, we can use sampling to approximate them by *simulating* such a random experiment and tracing the values of the random variables. Based on the simulation results we can then approximate the dependency values we would like to generate. For example, in our Random Experiment 2 one must first randomly pick a node and then a path from that node. One plausible sampling strategy is to first randomly put a mark on one node and then apply the method of *random walk* (i.e., from a node, randomly select an edge and walk to the target node, then perform the same experiment to reach farther nodes) (B. D. Hughes 1995) to walk through a path of certain length. Once a path is

generated this way, we then can record the starting node as one realization of random variable 1 and the path type as random variable 2. Repeating this kind of experiment will eventually create a set of sampling results and probability distribution on these two random variables, from which we can compute the contribution_2 and PMI_2 dependency measures.

It is clear now that an important benefit of choosing path types as features and exploiting random experiments to create feature values is that it *facilitates sampling*, which can significantly improve the efficiency and scalability of our system at the cost of a certain amount of accuracy. Questions such as how many samples are required for networks of different sizes and topologies to maintain a certain level of accuracy are beyond the scope of this thesis, but are undoubtedly an important and fruitful future research direction.

6.1.3 Outlier Ranking

Given the semantic profiles, the anomaly detection requires at most $O(n^2d)$ where n is the number of nodes and d is the size of the feature set. It is not the bottleneck of the system and there are some existing algorithms such as ORCA that can improve the performance to near linear time by randomization and pruning strategy (S. Bay and M. Schwabacher 2003).

6.2 Explanation-based Outlier Detection

This section describes one application for our explanation system, which we call the *Explanation-based Outlier Detection Algorithm*.

As described above, existing outlier algorithms aim at identifying abnormal points based on different criteria (i.e., distribution-based, clustering-based or distance-based). In this section we propose an alternative outlier-detecting criterion motivated by our explanation system, which we call explanation-based outliers.

Explanation-based outlier is based on the concept of minimum description length. It claims that *something is more abnormal if its uniqueness can be described with less effort or shorter description length*. Based on our explanation schema, a point can be regarded as more unusual if it is possible for a classifier to separate it from the others with fewer features. For example, in our problem, we can claim that a node is more abnormal if it can be explained by *fewer shorter path types*.

The general process for explanation-based outlier detection is simple. For each point in the data, the system first labels it as “abnormal” and the others as normal, and then applies a simple and non-opaque classifier such as decision tree to classify them. The depth of the resulting decision tree can be viewed as the description length of the point, and the system can then rank the abnormality of points based on this length. Note that one can also assign weights to different features based on their characteristics. For example, in our domain the features are path types, therefore, a feature of longer path length should be weighted more, since those generally lead to more complex explanations. That is to say, a 3-layer classification tree using paths of length four as features should have a longer description length compared with a 3-layer classification tree with paths of length three.

One major advantage of this algorithm compared with other outlier detection algorithms is that the explanations are automatically generated when the outlier is determined. Therefore, the users can easily understand why a point is chosen to be abnormal and how it is different from the others. The following is the algorithm for explanation-based outlier detection. Without loss of generality, we chose to return only the top abnormal point. In step 3, the *Des_length* function applies a decision tree classifier and returns the weighted description length of a point. Finally, the ArgMin function returns the point with the smallest length.

```
//P is an array of vectors, where each vector represents the feature values of a point
function Explanation_Based_Outliers (P)
1. array exp_length[|P|];
2.  for i:= 1 to |P|
3.      exp_length [i]=Des_length(P, P[i]);
4. return ArgMin_i(exp_length [i]);
```

6.3 Automatic Node Description

We now introduce a variation of our original anomaly detection task which we call *automatic node description*. The goal now is not to find abnormal nodes but instead to report important or interesting information about any given node. We believe this kind of system could be very useful for data analysis. For example, a police detective might want to learn something interesting or special about a person node they suspect, regardless of whether this person is abnormal or not; or a biologist

might want to find something special about a gene of interests. Doing this manually can be very hard and time consuming, because an MRN could contain many typed links and thousands of typed paths originating from a given node.

Fortunately, the method we designed to measure the node-path dependency can be of help. Note that based on the random experiments we designed, one can find path types that are highly correlated with a node such as those with high contribution, PMI or MI values. High PMI, for instance, implies that the path and node occur frequently together. It is not hard to imagine those types of paths are the ones we should report to the users. Moreover, our path-to-sentence generator can transform the paths into a more user-friendly form. The following is an algorithm for automatic node description supporting this idea:

```
//Input: G as the MRN, mc is a set of meta-constraints, n as the node to be explained, and
//k as the maximum path length
function describe_node (G, mc, n, k)
1. array P[] := extract_path_type (G, mc, n, k)
2.  for i:=1 to |P|
3.      array PMI [i]:=Get_PMI (G, P[i], n);}
4.  max_index := ArgMax_i(PMI [i]);
5. return path_2_sentence(P[max_index]);
```

For simplicity but without loss of generality, we assume PMI is used as the dependence measure. Also we chose to report only one feature with the highest dependency value. Note that the *extract_path_type* function in step 1 returns a set of

path types starting from *n*, satisfying *mc*, and has path length smaller than *k*. *Get_PMI* computes the PMI value of a given node with respect to a given path in the MRN.

The following are sample results generated by our system to explain three well-known individuals from the KDD Movie dataset: director *Oliver Stone*, actor *Meg Ryan*, and the movie *Gone with the Wind*. We exploit the relation-only meta-constraint to generate the semantic profile of a node, and then computed and translated the top two PMI_2 features for each node:

Olive Stone *Olive Stone wrote some movie adapted from a book whose author is also a composer (PMI=9.2)* *Oliver Stone produced some movie that has a composer who worked with some other movie person (PMI=6.5)*
Meg Ryan *Meg Ryan married to a movie person whose sibling is an actor (PMI=6.5)* *Meg Ryan produced some movie whose director wrote some book adapted into a movie (PMI=6.0)*
"Gone with the Wind" *"Gone with the Wind" has some actor who wrote a book adapted into a movie that was remade into another movie (PMI=6.6)* *"Gone with the Wind" is composed by somebody who is a cinematographer of a movie that has some director (PMI= 6.0)*

Besides the apparent application in homeland security and scientific discovery, the above results also suggest another interesting application for our system, which is to serve as a trivia question generator (e.g. "which movie has some actor who wrote a book adapted into a movie that was remade into another movie?").

6.4 Abnormal Path Discovery

Another variation of the original problem is to identify abnormal paths from the network. For example, an abnormal path between a suspicious person and a company might indicate some insider trading; or a chemist might have an interest in finding a special chain of reactions between two chemicals of interests.

More formally, given two nodes *x* and *y* in an MRN, we want to find a set of abnormal paths that connect them. To solve this problem we need to design a measure for path similarity and then find abnormal ones as those with fewer similar paths. We can apply the concept of meta-constraints described in Chapter 2 to determine whether paths are similar or not. That is, two paths are similar if they satisfy the same set of meta-constraints. For example, based on our default relation-only meta-constraint, the system can identify paths whose sequence of relations does not appear frequently as the abnormal ones. In (S. Lin and H. Chalupsky 2003a), we propose four different views (i.e., sets of meta-constraints) to determine whether paths are similar and then identify the rare ones as those of interest. The following is the algorithm to find the most abnormal path between two given nodes. The *extract_path* function first returns a set of paths starting from *s* and ending at *t*, satisfying *mc* and with path length smaller than *k*. The s*im* function counts how many similar paths (based on the meta-constraint set) there are for a given one. Finally, it returns the natural language representation for the rarest path.

```
//Input G is the MRN, mc is a set of meta-constraints, s and t are two given nodes, and k is
//the maximum path length
function Path_discover (G, mc, s, t, k)
1. array P[] := extract_path (G, mc, s, t, k)
2. array abn [|P|];
3.  for i:=1 to |P|
4.       abn[i]=1/ sim(G, s, t, P[i], mc);
5. abn_index= ArgMax_i(abn [i]);
6. return path_2_sentence(P [abn_index]);
```

In (S. Lin and H. Chalupsky 2003b) we conducted experiments to find abnormal paths in the HEP-Th bibliography network. We found several interesting results. The first type of rare path our system found was what we called a citation loop: *"paper x cites paper y cites paper z cites paper x"*. In fact this is somewhat of a contradiction because for one paper to cite another, the cited one has to be published earlier then the citing paper. Therefore this loop should normally not occur. We found that one of the reasons this can happen in the HEP-Th dataset is that it is an archive that allows people to withdraw and update their publications. Therefore, it is possible to update an existing paper by adding citations to papers that were published after the original publication of the paper. Two other abnormal and interesting paths we found are *"a paper cites itself"* and *"an author published multiple papers at the same time"*. The former is interesting because it might indicate an error, and the latter tells us that in

the in the area of high energy physics theory, researchers in general do not publish multiple papers at the same time.

6.5 Usage Scenarios and Usefulness of Results

Since UNICORN is a data-driven system, the potential interestingness and usefulness of the results are highly correlated with the quality of the data—that is, whether the information encoded in the network is relevant and sufficient—as well as the goal of the task. This section discusses the usability of UNICORN and its explanation system, specifically the circumstances that lend themselves best to the application of UNICORN for analysis.

6.5.1 Applying UNICORN for seed identification

A first usage scenario for UNICORN is to apply it in the early stage of network analysis. In a large network with thousands or millions of nodes and labeled links, it is not trivial for a human to know how and where to start analyzing the data. One plausible solution is to apply UNICORN and its explanation system to first identify and explain the few most abnormal nodes in the network. The users can choose these nodes as seeds and perform further analysis starting from them. Such preprocessing is also very helpful for some group detection systems such as (J. Adibi, H. Chalupsky et al. 2004) which require a set of given seeds to grow the groups.

6.5.2 Applying UNICORN for goal-oriented node identification tasks

Theoretically, UNICORN identifies instances which play abnormal roles (or surrounded by abnormal labeled network structure) in the network. Just like any other data-driven discovery system (for example, association rule mining), it is generally

hard to predict whether the system can find things that can match a predefined goal (other than finding abnormal things). Whether the finding is useful for a particular task depends on not only the content and quality of the data but also the users' own subjective, internal interpretation of interestingness or usefulness. Therefore in the thesis we do not claim that *all* the instances UNICORN finds are useful. Instead we show that UNICORN can find nodes that highly correspond to the suspicious ones (as we discuss in Chapter 3), and sometimes interesting (as the results in Chapter 5). In Chapter 4 we further demonstrate that the semantic profiles generated by UNICORN are useful and meaningful since they can assist the users to determine the crime organizers. Our experiments can be regarded not only as the evaluation of our algorithm but also the demonstration of how and in which circumstance one should apply UNICORN for data analysis. The results indicate that utilizing the surrounding labeled network structure to represent the meaning of the nodes and then identifying nodes with abnormal meanings is a proper strategy for finding suspicious and interesting things in certain domains.

6.5.3 UNICORN as a semantic interpretation system for MRN's

This section suggests how UNICORN should be applied in the situation when we are not sure whether the targets to be found possess abnormal property or whether the information provided by the network is not skewed (if it is, then there is a high chance the skewed individual such as Hitchcock in the KDD Movie dataset will be regarded as abnormal). For these situations we recommend exploiting UNICORN as a semantic interpretation tool instead of a knowledge discovery tool. That is, instead

of identifying abnormal instances, we apply UNICORN to generate an alternative interpretation of the data and let users to decide whether the instacnes are interesting or not.

For this purpose, we propose to modify UNICORN slightly by removing the distance-based outlier finder and have the explanation system to explain *every* node in the network based on its semantic profile. As a result, instead of telling users which node is abnormal, we simply provide an alternative view of the network as a list of node descriptions (descriptions can be ranked by the length of the explanations, similar to the idea of explanation-based outliers) showing what makes each node unique. Remember that the results of our human study described in Chapter 4 reveal that it is really hard for humans to manually analyze an MRN to identify certain type of instances (e.g. suspicious or abnormal ones). In this case, we believe by transforming and condensing this MRN into a list of user-friendly descriptions about the nodes, the users will have higher chance to find the individuals they are interested in. Our human study demonstrates an alternative way UNICORN can be applied, and the results assures that after transforming the original network into a set of natural language descriptions of the nodes as shown in Appendix III, human subjects can perform their tasks much more efficiently and accurately.

Chapter 7

Related Work

Our problems and solutions are related to a variety of research fields including knowledge discovery and data mining, scientific discovery, social network analysis and machine learning. For each related field, we will describe its definition, main goals, methodologies and its similarity as well as differences to our research.

7.1 Network Analysis for Homeland Security and Crime

Our task is related to crime or homeland security network analysis in the sense that one major application for our system is to identify suspicious and abnormal individuals from data. There has been a variety of research focusing on applying intelligent graph analysis methods to solve problems in homeland security (D. Jensen, M. Rattigan et al. 2003; R. Popp, T. Armour et al. 2004; H. Chen and F. Wang 2005) and crime mining (J. Xu and H. Chen 2005; H. Chen and J.Xu 2006). Adibi et al. describe a method combining a knowledge-based system with mutual information analysis to identify groups in a semantic graph based on a set of given seeds (J. Adibi, H. Chalupsky et al. 2004). Krebs describes a social network analysis approach on the 9/11 terrorists network and suggests that to identify covert individuals it is preferable to utilize multiple types of relational information to uncover the hidden connections in evidence (V. Krebs 2001). This conclusion echoes our decision of performing

discovery on top of a multi-relational network. There are also link discovery and analysis algorithms proposed to predict missing links in graphs or relational datasets (B. Taskar, M. Wong et al. 2003; S. F. Adafre and M. de Rijke 2005). Recently, several general frameworks have been proposed to model and analyze semantic graphs such as relational Bayesian networks, relational Markov networks, and relational dependency networks (M. Jaeger 1997; R. Bunescu and R. Mooney 2004; J. Neville and D. Jensen 2004). However, these frameworks aim at exploiting the graph structure to learn the joint or posterior probabilities of events or relations between them based on training examples. Our task and goal is different—we are not assuming there is a given or trainable causal or Bayesian network and we are focusing on identifying abnormal instances in an unsupervised manner. Furthermore, our framework has the ultimate goal of being able to generate human understandable explanations for its findings.

7.2 Social Network Analysis

Social networks consist of a finite set of actors (nodes) and the binary ties (links) defined between them. The actors are social elements such as people and organizations while the ties can be various types of relationships between actors such as biological relationships or behavior interactions. The goal of social network analysis (SNA) is, stated briefly, to provide better understanding about the structure of a given social network (S. Wasserman and K. Faust 1994). Although most of the analyses are focused on finding social patterns and subgroups, there are a small number of SNA tasks resembling our instance discovery problem. *Centrality analysis*

aims at identifying important nodes in the network based on their connectivity with others: An actor is important if it possesses high node degree (degree centrality) or is close to other nodes (closeness centrality). An actor is importantly connected to two source actors if it is involved in many connections between them (betweenness and information centrality). The major difference between centrality analysis and our approach is that centrality looks for central or highly connected nodes, while our system looks for those that are different from others. *Social positions analysis* targets finding individuals who are similarly embedded in networks of relations. The similarity between actors is measured by whether they have similar ties with other actors. In UNICORN, we generalize this concept by using paths and their statistical contributions. The generalized path features and their dependency measures allow us to exploit more information from the network in order to capture the deeper meaning of instances.

Another major difference between the problems SNA handles and our problem is that most of the SNA approaches are designed to handle only one-mode or two-mode networks (i.e., there are at most two types of actors), while such a limitation does not exist in our analysis. Moreover, existing statistical and graph-theoretical methods for centrality, position, and role analysis do not distinguish between different link types and their different semantics. The relational network we are dealing with is also not limited to social networks–the networks we focus on can be any relational graph (even, for instance, a thesaurus such as WordNet).

7.3 Knowledge Discovery and Data Mining

Knowledge discovery and data mining (KDD) research focuses on discovering and extracting previously unknown, valid, novel, potentially useful and understandable patterns from lower-level data (U. Fayyad, G. Piatetsky-Shapiro et al. 1996). Such patterns can be represented as association rules, classification rules, clusters, sequential patterns, time series, contingency tables, etc (R. Hilderman and H. Hamilton 1999). Below we describe KDD research most relevant to our problems.

7.3.1 Graph Mining

Graph mining aims at mining data represented in graphs or trees, such as mining interesting subclasses in a graph (T. Murata 2003; J. Ramon and T. Gaertner 2003) and mining the Web as a graph (J.M. Kleinberg, R. Kumar et al. 1999; R. Kumar, P. Raghavan et al. 2000). It is similar to our problem in the sense that a network is a type of graph. There are several well-known objective methods (e.g. PageRank) that discover important nodes in a graph. However, to our knowledge there is no work addressing how to determine interesting or abnormal instances in graphs.

That being said, there are various findings about the nature of networks that are worth mentioning: *Small world analysis* (J. Kleinberg. 2000) shows that with a small amount of links connecting major clusters, it is essentially possible to link any two arbitrary nodes in a network with a fairly short path. *The strength of weak ties* approach (M. Granovetter 1973) addresses the issue that weak connections between individuals might be more important than strong ones, because they act like bridges. This concept is to some extent similar to our abnormal path discovery in the sense

that rare paths also represent a kind of weak connection given a specific similarity measure. The major difference between our approach and the approaches described above is that our system not only models complex syntactic structure of the typed graphs but also incorporates statistical dependency measures to capture deeper meanings of the nodes.

7.3.2 Relational Data Mining

Relational data mining (RDM) deals with relational tables in a database. It is related to our problem in the sense that a multi-relational network is a type of relational data and can be translated into relational tables. RDM searches a language of relational patterns to find patterns that are valid in a given relational database (S. Dzeroski 2001). Morik proposes a way to find interesting instances in this relational domain by first learning rules and than searching for the instances that satisfy one of the following three criteria: exceptions to an accepted given rule; not being covered by any rule; or negative examples that prevent the acceptance of a rule (K. Morik 2002). Angiulli propose similar ideas by using default logic to screen out the outliers (F. Angiulli, R. Ben-Eliyahu-Zohary et al. 2003). Both methods require a certain amount of domain knowledge or training example for supervised learning. It is different from our system since we are looking for abnormal instances that are possibly neither expected by users nor biased by training data. There is one type of unsupervised discovery problem called interesting subgroup discovery that tries to discover subsets that are unusual (W. Klosgen 1996; S. Wrobel 1997; P. Flach and N. Lachiche 2001). For example, interesting subsets are those whose distribution of

instances based on a certain objective function is different from that of the whole dataset. The major difference between our problem and subgroup discovery is that we are not looking for groups or patterns but individual instances.

Inductive logic programming (ILP) is one of the most popular RDM methods whose goal is mainly to induce a logic program corresponding to a set of given observations and background knowledge represented in logic form (S. Dzeroski and N. Lavrac 2001). ILP has been successfully applied to discover novel theories in various science domains such as math (S. Colton and S.H. Muggleton. 2003) and biology (S. Colton 2002). The standard ILP problem is not similar to ours, because it works in a completely supervised manner. However, there is an extension of ILP that exploits a distance-based measure among relational instances (U. Bohnebeck, T. Horvath et al. 1998; T. Horvath, S. Wrobel et al. 2001; M. Kirsten, S. Wrobel et al. 2001) that is closely related to our problem. Their work defines distance between propositional instances as the accumulation of the distances for all their attributes. If the attribute is numerical, the distance can be defined as the difference between two attributes divided by the maximum distance. If an attribute is discrete (categorical), then the distance is 1 if they have different values and 0 if identical. However, to apply such an approach to our problem, one needs to first clearly define the common and representative features (e.g. semantic profiles in UNICORN) for instances, which leads to another research problem called propositionalization.

7.4 Propositionalization

Transforming a relational representation of a learning problem into a propositional (feature-based attribute-value, single table) representation is generally called propositionalization (S. Kramer, N. Lavrac et al. 2001). The benefit of mapping relational data into propositional data is that propositional learning is one the most general and best studied fields in data mining. Propositionalization can be regarded as a feature construction procedure that tries to create a small but representative feature set based on the conjunction of literals, and consequently, the values of the features are boolean. One popular idea of feature construction is to find the ones that are shared by a statistically significantly large fraction of the training set. For example, WARMR tries to detect frequently succeeding queries as features (L. Dehaspe and H. Toivonen 1999), and Kramer and Frank propose a bottom-up approach method to determine frequently occurring fragments as new features (S. Kramer and E. Frank 2000). There are other general methods proposed for propositionalization: The Linus system uses background knowledge to generate the individual facts about the target relation, and then induces the target relation through a propositional learning algorithm (N. Lavrac, S. Dzeroski et al. 1991). Kramer et al. use minimum description length as a fitness function to search for features that work well together in terms of describing the data (S. Kramer, B. Pfahringer et al. 1998). In our design, we do not weight the frequency or description length of the features because we believe both the frequent and infrequent features have the potential to

convey important information for anomaly detection. Instead we design the meta-constraints which allow the system to systematically select a set of features.

To classify nodes, Geibel and Wysotzki propose to obtain features in the graph by analyzing categories of paths in the graph (P. Geibel and F. Wysotzki 1995; P. Geibel and F. Wysotzki 1996). Instances can then be represented by boolean path features describing whether they are involved in various types of paths. The idea is similar to our idea of generating path types in the graph as features. However, instead of applying symbolic boolean feature values, we take a step further by incorporating the statistical measures (e.g. MI) to capture deeper dependency information in the network, which allows us to better capture the semantics of the instances and avoids the overfitting problems.

7.5 Interestingness Measurements

In data mining and machine learning, various efforts have been made to define subjective or objective interestingness measurements for different tasks. In this section, we would like to investigate whether those measurements can be exploited to discover abnormal instances in a multi-relational network.

Based on subjective measures, one can define two different types of interesting discoveries: the first are those that bring completely new knowledge about a domain and the second are those contradicting a user's belief (A. Silberschatz and A. Tuzhilin 1995; B. Padmanabhan and A. Tuzhilin 1999). However, to do such, we either need some domain knowledge or information about the user's belief, both of which are not available in our problem.

As for objective measures, the rule-interest function (G. Piatetski-Shapiro 1991) proposes that a rule $X \rightarrow Y$ is interesting if the co-occurrence rate of *X* and *Y* is much higher than expected if they were independent. This idea was later extended to encompass *confidence* (given a transactions which contains *X* the probability that it also contains *Y*) and *support* (the percentage of transactions that contains both *X* and *Y* items) for mining interesting association rules (R. Agrawal, T. Imielinski et al. 1993). The idea of *deviation* has also been used to measure interestingness (G. Piatetsky-Shapiro and C. J. Matheus 1994) by identifying a large deviation between an observed value to a reference value (e.g. previous value). Kamber and Shinghal propose to use *sufficiency* (the probability that the evidence occurs given the hypothesis is true divided by the probability of the evidence given that the hypothesis is false) and *necessity* (the probability that the evidence does not occur given the hypothesis is true divided by the probability that the evidence does not occur given that the hypothesis is false) to rank the interestingness of hypotheses. (R. Hilderman and H. Hamilton 1999) proposes several statistical and information-theoretic heuristics (*average information content* and *total information content*, for example) to measure the interestingness of a summary of data. The above measures are different from our method, because they are not exploiting the concept of relative abnormality (or distance-based outlier), and they are usually only applicable to measure the interestingness of a rule or pattern instead of an instance such as a node in a relational network.

Even so, there is a significant amount of work on *unexpectedness* measures. Objective unexpectedness measures, in a nutshell, regard discovered results to be surprising if they are different from other candidates based on certain distance metrics (G. Dong and J. Li. 1998; P. Gago and C. Bento 1998; N. Zhong, Y. Yao et al. 2003). In addition, Freitas points out three alternative unexpectedness indicators that are worth noticing: "small disjuncts" (rules whose coverage is small); rules whose antecedent contains attributes with low information-gain; and the occurrence of Simpson's paradox (that is, an association between a pair of variables can consistently be inverted in each subpopulation when the population is partitioned (A. A. Freitas 1998). Although these measures are (to some extent) similar in spirit to the distance-based outlier, they are designed for mining propositional numeric datasets and it is not clear how they could be applied to a multi-relational network.

The research of anomaly detection in computer security also attempts to find abnormal behavior. It quantifies usual behavior and flags other irregular behavior as potentially intrusive (S. Kumar 1995). The basic idea of anomaly detection is to first construct opinions on what is normal, either by learning (H. Debar, M. Becker et al. 1992) or a rule-based system (B. Mukherjee, L. T. Heberlein et al. 1994) , and then flag behaviors that are unlikely to originate from the normal process (W. Lee and S. Stolfo 1998). In anomaly detection, the target domain and data are usually well defined in numerical values, and the focus is on finding meaningful patterns and, consequently, anomaly efficiently (e.g. in real time). With our problem, by contrast, the major difficulty lies in how to model the *relational network data* into a form that

facilitates the discovery of anomalies as well as their explanations. Furthermore, generating explanation is not a major focus of the above systems. UNICORN can be viewed as an anomaly detection system in a non-numerical semantic graph with embedded explanation capability.

7.6 Outliers

Below we briefly review the literature on outlier detection, because we not only apply an outlier detector in our discovery system but also need to describe why an outlier is abnormal in the explanation system.

7.6.1 Outlier Definition and Detection:

An outlier is an observation that deviates so much from other observations to arouse suspicion that it was generated by a different mechanism (D. Hawkins 1980). Outlier detection is an important technology with a variety of applications such as video surveillance and fraud detection.

There are three major groups of outliers and each has its associated detection algorithm: clustering-based, distribution-based and distance-based outliers. Clustering-based outliers are the points that cannot be clustered into any clusters. Clustering systems such as CLARANS (R.T. Ng and J. Han 1994), BIRCH (T. Zhang, R. Ramakrishnan et al. 1996) and CLUE (S. Guha, R. Rastogi et al. 1998) extract outliers as the by-products of clustering. FindCBLOF tries to rank a local clustering-based outlier not only by how far an “outlier cluster” is away from the nearest cluster, but also by how large its size is (H. Zengyou, X. Xiaofei et al. 2003). The problem with clustering-based outliers is that in order to find the outliers, one must first define

a similarity measure and produces clusters. This could cause problems for datasets that are hard to cluster.

Distribution-based outlier detection, on the other hand, takes the known statistical distribution of the data and identifies deviating points as outliers (V. Barnett and T. Lewis 1994). The problem with distribution-based outlier detection is that there is no guarantee that the underlying data distribution is always accessible or learnable. This is especially true for large, high-dimensional datasets.

To deal with these drawbacks, Knorr and Ng propose distance-based outliers that facilitate outlier detection without having to be given a priori knowledge about the distribution of the data (E. Knorr and R. Ng 1998). Their idea is simple: a point *x* in a data set is an outlier with respect to parameters *n* and *d* if no more than *n* points in the data set are at a distance *d* or less from *x*. The major problem of Knorr and Ng's algorithm, as addressed by Ramaswamy and many others, is that it is not clear how to determine the threshold distance *d*. Ramaswamy et al. propose a modification to use the distance to its *kth-nearest neighbor* to rank the outlier (S. Ramaswamy, R. Rastogi et al. 2000). In other words the top *n* points with the largest *kth*-neighbor distance are outliers. Later, Breunig, et al., develop this idea further, proposing the idea of density-based local outlier that enables the system to quantify how outlying a point is (M. M. Breunig, H. Kriegel et al. 2000). A local outlier, which is different from the "global" outliers, is a point that possesses sparser density than its neighborhood points (i.e., the number of points surrounding it is much smaller than those of its neighbors).

The outlier detection algorithms described above only target numerical datasets. However, there is some recent work on outlier detection for non-numerical data. One worth mentioning is Wei's HOT (Hypergraph-based Outlier Test) system for outlier detection in categorical data (L. Wei, W. Qian et al. 2003). He proposes to extract frequent itemsets first and then defines outliers as those items belonging to a frequent itemset but that have certain attributes that look different. For example, the item "a sedan car owned by a young person whose salary is high" is an outlier because young rich persons (a frequent itemset) usually have sports cars. The idea is similar to the concept of local outlier, because it first defines a local set (a frequent itemset) and then looks for individuals that are abnormal in its local area. It is different from our works because we aim to process relational network data instead of categorical data.

7.6.2 Outlier Explanation

There is a small amount of prior work on outlier explanation. For distribution-based outliers, Yamanishi and Takeuchi propose to combine statistical methods with supervised methods to generate outliers (K. Yamanishi and J. Takeuchi 2001). The statistical method is first applied to learn the data distribution and then to identify the outliers. Once the outliers are detected, the classification method can be applied to extract the filtering rules as explanation. The idea of applying a classification method for explanation is similar to our explanation system. Our system improves upon this by introducing a variety of different explanation schemas (e.g. adding a reference class) for different types of networks, and furthermore translates the results into natural language. Moreover, in the explanation-generation stage Yamanishi and

Takeuchi classify all outliers in a single shot. This approach is suitable for a situation where the distribution is known, but not for a distance-based scenario in which the outliers could be very diverse and each of them might require a different reference class for explanation generation. Yao et al. propose to apply a classification method to generate explanations for association rules (Y. Yao, Y. Zhao et al. 2003). Their system utilizes external information that was not used in association-rule mining to generate the condition in which the rules hold. The external information is not accessible to our program, so it has to rely on internal information only to produce explanations.

7.7 Machine Discovery in Science and Literature-based Discovery

Machine discovery in science has been an important research area in AI for more than twenty years. Herbert Simon described it as "gradual problem-solving processes of searching large problem spaces for incompletely defined goal objects" (H. Simon 1995). The majority of machine discovery programs focus on discovering (or rediscovering) the theories and laws of natural science which can be viewed as search for a parsimonious description of the world (A. Milosavljevic 1995). Langley (P. Langley 1998) classifies scientific discovery into five stages where various systems have been developed to achieve each of them: finding taxonomies (e.g. clustering systems), finding qualitative laws (R. Jones 1986), finding quantitative laws (P. Langley, G.L. Bradshaw et al. 1981), developing structure models (J.M. Zytkow 1996), and process models (R. Vlades-Perez 1995). Scientific discovery systems have been applied to areas such as mathematics, physics, chemistry,

linguistics, etc. AM is a heuristic-based search system for mathematical laws and concepts such as natural numbers and Goldbach's conjecture (D. Lenat 1982). GRAFFITI (S. Fajtlowicz 1988) has successfully generated hundreds of conjectures about inequalities in graph theory by heuristic search, many of which lead to publications when mathematicians tried to prove or refute these conjectures. MECHEM (R.E. Valdes-Perez 1995) is a discovery tool that hypothesizes the structural transformations of chemicals. (R.E. Valdes-Perez and V. Pericliev 1999) justify their maximally parsimonious discrimination program by rediscovering several linguistic phenomena such as the structure of kinship-related terms in Seneca (originally found by Lounsbury in 1964) as well as Greenberg's language universal rules (V. Pericliev 2002). There is another research branch called literature-based discovery, which aims at finding interesting connections among concepts in different documents. ARROWSMITH (N. Smalheiser and D. Swanson 1998) is a literature-based discovery tool that hypothesizes possible treatments or causes of diseases using a collection of titles and abstracts from the medical literature. Unlike other scientific discovery programs which focus on discovering theories or laws, ARROWSMITH focuses on instance discovery.

These scientific discovery systems are similar to our research in the sense that researchers also propose a series of heuristics to discover interesting instances in their domain, and our system can as well be applied for this purpose. However, most of the scientific discovery systems are knowledge driven, which means the developer has to first provide some background information as well as an underlying model while the

major challenge lies in how to efficiently search for the solution. The major challenge for our problem, in contrast, is not merely about search but also how to automatically model the problem into something that allows us to find meaningful results as well as explain them. Technically speaking, most scientific discovery systems shift the responsibility of feature construction and explanation generation to the developers or users and focus on search, while our discovery system tries to complete the whole discovery process from modeling to search to explanation. The other differentiating feature of our system is that, unlike other discovery programs that focus on a certain domain, it does not restrict itself to one specific realm as long as an MRN describing the data can be provided. We hope the design of our system can provide some fresh new ideas about how one can develop a more general framework to assist scientific discovery.

Chapter 8

Conclusion

In this dissertation we described a general unsupervised framework for identifying abnormal nodes in large and complex multi-relational networks and an explanation mechanism to explain the discovered results.

The first contribution is the design and development of a novel framework called UNICORN that integrates symbolic and statistical methods to capture the syntactic semantics of the nodes into “semantic profiles”. Given these semantic profiles we can compare and contrast nodes and exploit distance-based outlier detection to identify abnormal nodes. Since the method is unsupervised and does not require training examples or user-defined features, it has the potential to discover truly novel instances without being biased by human analysts or training examples. Our experiments show that UNICORN can successfully identify suspicious crime organizers and that it outperforms other network algorithms that were applied to analyze the 9/11 terrorist network by a large margin. We also discuss the computational complexity of UNICORN and show how an incremental algorithm as well as sampling technologies can address them. One promising future direction is to utilize our dependency measure and semantic profiles of nodes for other purposes such as node clustering in graph datasets.

Motivated by issues of verification and the danger posed by false positives that might mistakenly incriminate innocent individuals, our system needs to not only identify abnormal nodes but also be able to explain to an analyst why they were abnormal. To this end we designed and implemented a novel explanation mechanism for UNICORN that can produce four different types of explanations. To generate its explanations, UNICORN selects a small set of features that account for the abnormality of the node to be explained, and summarizes them in a human-understandable form such as natural language. In an experiment with human subjects, we demonstrate that the explanation mechanism can significantly improve the accuracy and efficiency of subjects when analyzing a complex dataset. One promising future direction regarding the explanation-based discovery framework is to generalize the explanation mechanism for other knowledge discovery processes such as rule-mining or clustering.

Through our experiments performed on two representative natural datasets in the movie and bibliography domains, we show that the UNICORN framework is domain independent and can be applied not only to identify suspicious instances in crime datasets, but also to find and explain abnormal or interesting instances in any multi-relational network. This leads to potential applications in a variety of areas such as scientific discovery, data analysis and data cleaning. Due to the generality of the techniques we developed they also lend themselves to other applications such as a novel outlier detection mechanism called *explanation-based outlier detection*, general node explanation to describe pertinent characteristics of arbitrary nodes, and

abnormal path discovery to detect abnormal paths between nodes. In the future, we would like to seek further collaborations with scientists from other areas such as microbiology, chemistry, pharmacology, medicine, etc. to apply UNICORN for scientific discovery in their domain of interest.

We conclude with two other possibilities for important future research directions. The first pertains to knowledge representation for an MRN: from a given dataset, how can one determine which information is important and useful and how this information should be connected and represented to generate the MRN. This is an important problem, since the construction of the network can have a lot of impact on the results our system can discover. The second problem is to provide UNICORN with the capability of dealing with temporal information, as information on the way things change over time can doubtless lead to abnormal and interesting findings.

We believe performing knowledge discovery in large, heterogeneous networks with a variety of different types of relations is an important new research direction with many potential applications. By reporting our methods and results in this thesis as one of the pioneer efforts to deal with abnormal instance discovery, we hope to draw more attention and motivate further ideas in this research domain.

References

S. F. Adafre and M. d. Rijke (2005). "Discovering missing links in wikipedia." In Proceedings of the Workshop on Link Discovery: Issues, Approaches and Applications (LinkKDD-2005).

J. Adibi, H. Chalupsky, E. Melz and A. Valente (2004). "The KOJAK group finder: connecting the dots via integrated knowledge-based and statistical reasoning." In Proceedings of the Innovative Applications of Artificial Intelligence (IAAI), p.800-807.

R. Agrawal, T. Imielinski and A. Swami (1993). "Mining association rules between sets of items in large databases." In Proceedings of the ACM Special Interest Group on Management of Data(SIGMOD), p.207-216.

F. Angiulli, R. Ben-Eliyahu-Zohary and L. Palopoli (2003). "Outlier detection using default logic." In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), p.833-838.

V. Barnett and T. Lewis (1994). "Outliers in statistical data." New York, John Wiley and Sons.

S. Bay and M. Schwabacher (2003). "Mining distance-based outliers in near linear time with randomization and a simple pruning rule." In Proceedings of the International Conference on Knowledge Discovery and Data Mining, p.29-38.

U. Bohnebeck, T. Horvath and S. Wrobel (1998). "Term comparisons in first-order similarity measures." In Proceedings of the 8th International Conference on Inductive Logic Programming, p.65-79.

M. M. Breunig, H. Kriegel, R. T. Ng and J. Sander (2000). "LOF: Identifying density-based local outliers." In Proceedings of the ACM SIGMOD International Conference on Management of Data, p.93-104.

S. Brin and L. Page (1998). "The anatomy of a large-scale hypertextual Web search engine." Computer Networks and ISDN Systems 30: p.107-117.

R. Bunescu and R. Mooney (2004). "Relational Markov networks for collective information extraction." In Proceedings of the International Conference on Machine Learning (ICML).

H. Chen and J.Xu (2006). "Intelligence and security informatics." Annual Review of Information Science and Technology 40: p.229-289.

H. Chen and F. Wang (2005). "Artificial Intelligence for Homeland Security." IEEE Computer Society: p.12-16.

S. Colton (2002). "Automated theory formation applied to mutagenesis data." In Proceedings of the First British Cuban Workshop on BioInformatics.

S. Colton and S. H. Muggleton. (2003). "ILP for mathematical discovery." In Proceedings of the 13th International Conference on Inductive Logic Programming, p.93-111.

H. Debar, M. Becker and D. Siboni (1992). "A neural network component for an intrusion detection system." In Proceedings of the IEEE Computer Sociecty Symposium on Research in Security and Privacy, p.240-250.

L. Dehaspe and H. Toivonen (1999). "Discovery of frequent datalog patterns." Data Mining and Knowledge Discovery 3(1): p.7-36.

J. S. Dhaliwal and I. Benbasat (1996). "The use and effects of knowledge-based system explanations: theoretical foundations and a framework for empirical valuation." Information Systems Research 7: p.342-362.

G. Dong and J. Li. (1998). "Interestingness of discovered association rules in terms of neighborhoodbased unexpectedness." In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Melbourne, Australia, p.72-86.

S. Dzeroski (2001). Data mining in a nutshell. Book "Data mining in a nutshell". S. Dzeroski and N. Lavrac. New York, Springer-Verlag**:** 1-27.

S. Dzeroski and N. Lavrac (2001). An introduction to inductive logic programming. Book "An introduction to inductive logic programming". S. Dzeroski and N. Lavrac, Springer-Verlag**:** 48-73.

S. Fajtlowicz (1988). "On conjectures of Graffiti." Discrete Mathematics 72: p.113-118.

U. Fayyad, G. Piatetsky-Shapiro and P. Smyth (1996). "The KDD process for extracting useful knowledge from volumes of data." Communications of the ACM 39(11): p.27-34.

P. Flach and N. Lachiche (2001). "Confirmation-guided discovery of first-order rules with Tertius." Machine Learning 42(1-2): p.61-95.

A. A. Freitas (1998). "On objective measures of rule surprisingness." In Proceedings of the Second European Conference on the Principles of Data Mining and Knowledge Discovery (PKDD), p.1-9.

P. Gago and C. Bento (1998). "A Metric for selection of the most promising rules." In Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery, p.19-27.

P. Geibel and F. Wysotzki (1995). "Learning context dependent concepts." In Proceedings of the 5th International Workshop on Inductive Logic Programming, p.323-337.

P. Geibel and F. Wysotzki (1996). "Relational learning with decision trees." In Proceedings of the European Conference on Artificial Intelligence, p.428-432.

M. Granovetter (1973). "The strength of weak ties." American Journal of Sociology 78: p.1360-1380.

S. Guha, R. Rastogi and K. Shim (1998). "Cure: An efficient clustering algorithm for large databases." In Proceedings of the ACM Special Interest Group on Management of Data(SIGMOD), p.73-84.

D. Hawkins (1980). "Identification of outliers." London, Chapman and Hall.

S. R. Haynes (2001). "Explanation in Information Systems: A Design Rationale Approach." The London School of Economics.

R. Hilderman and H. Hamilton (1999). Knowledge discovery and interestingness measures: A survey. Technical Report CS 99-04, Department of Computer Science, University of Regina.

R. K. Hill (1995). "Non-well-founded set theory and the circular semantics of semantic networks." In Proceedings of the Intelligent Systems: Third Golden West International Conference, Dordrecht, Netherlands, Kluwer Academic Publishers, p.375-386.

T. Horvath, S. Wrobel and U. Bohnebeck (2001). "Relational instance-based learning with lists and terms." Machine Learning 43: p.53-80.

B. D. Hughes (1995). "Random walks and random environments." Oxford University Press.

M. Jaeger (1997). "Relational Bayesian networks." In Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence, p.266-273.

D. Jensen, M. Rattigan and H. Blau (2003). "Information awareness: A prospective technical assessment." In Proceedings of the Ninth ACM International Conference on Knowledge Discovery and Data Mining, Washington, DC, p.378-387.

I. Jolliffe (1986). "Principle component analysis." Journal of Educational Psychology 24: p.417-441.

R. Jones (1986). "Generating predictions to aid the scientific discovery process." In Proceedings of the Fifth National Conference of the American Association for Artificial Intelligence, p.513-517.

M. Kirsten, S. Wrobel and T. Horvath (2001). Distance based approaches to relational learning and clustering. Book "Distance based approaches to relational learning and clustering". S. Dzeroski and N. Lavrac, Springer-Verlag**:** 213-232.

J. M. Kleinberg (1999). "Authoritative sources in a hyperlinked environment." Journal of the ACM 46(5): p.604-632.

J. M. Kleinberg (2000). "The Small-World phenomenon: an algorithmic perspective." Nature 406(6798).

J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins (1999). "The Web as a Graph: Measurements, Models, and Methods." In Proceedings of the 5th International Conference on Computing and Combinatorics, Japan, p.26-28.

J. Kleinberg. (2000). "The Small-World Phenomenon: An Algorithmic Perspective." Nature 406(6798).

W. Klosgen (1996). Explora: a multipattern and multistrategy discovery assistant. Book "Explora: a multipattern and multistrategy discovery assistant". U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, AAAI/MIT Press**:** 249-271.

E. Knorr and R. Ng (1998). "Algorithms for mining distance-based outliers in large datasets." In Proceedings of the International Conference on Very Large Data Bases, p.392-403.

S. Kramer and E. Frank (2000). "Bottom-up propositionalization." In Proceedings of the 10th International Conference on Inductive Logic Programming, p.156-162.

S. Kramer, N. Lavrac and P. A. Flach (2001). Propositionalization approaches to relational data mining. Book "Propositionalization approaches to relational data mining". N. Lavrac and S. Dzeroski, Springer.

S. Kramer, B. Pfahringer and C. Helma (1998). "Stochastic propositionalization of non-determinate background knowledge." Lecture Notes in Artificial Intelligence 1446: p.80-94.

V. Krebs (2001). "Mapping networks of terrorist cells." Connections 24(3): p.43-52.

R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins and E. Upfal (2000). "Stochastic models for the Web graph." In Proceedings of the 41st IEEE Annual Foundations of Computer Science, p.57-65.

S. Kumar (1995). "Classification and detection of computer intrusions." Ph.D., Computer Sciences, Purdue, W. Lafayette, IN

P. Langley (1998). "The computer-aided discovery of scientific knowledge." In Proceedings of the First International Conference on Discovery Science, Fukuoka, Japan.

P. Langley, G. L. Bradshaw and H. A. Simon (1981). "Bacon.5: The discovery of conservation laws." In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, p.121-126.

N. Lavrac, S. Dzeroski and M. Grobelnik (1991). "Learning nonrecursive definitions of relations with LINUS." Lecture Notes in Artificial Intelligence 482: p.265--281.

W. Lee and S. Stolfo (1998). "Data mining approaches for intrusion detection." In Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, p.79-94.

D. Lenat (1982). "The nature of heuristics." Artificial Intelligence 19: p.189-249.

S. Lin (2006). "Generating natural language descirption for paths in the semantic network." master final project report, USC Linguistics Department,

S. Lin and H. Chalupsky (2003a). "Unsupervised link discovery in multi-relational data via rarity analysis." In Proceedings of the IEEE International Conference on Data Mining (ICDM), Florida, p.171.

S. Lin and H. Chalupsky (2003b). "Using unsupervised link discovery methods to find interesting facts and connections in a bibliography dataset." KDD Explorations 5(2): p.173-179.

S. Lin and H. Chalupsky (2004). "Issues of verification for unsupervised discovery systems." In Proceedings of the KDD Workshop on Link Discovery, Seattle.

A. Milosavljevic (1995). "The discovery process as a search for concise encoding of observed data." Foundations of Scinece 2: p.201-224.

K. Morik (2002). "Detecting interesting instances." In Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery, London, UK, p.13-23.

B. Mukherjee, L. T. Heberlein and K. N. Levitt (1994). "Network intrusion detection." IEEE Network 8(3): p.26-41.

T. Murata (2003). "Graph mining approaches for the discovery of Web communities." In Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences (MGTS-2003).

J. Neville and D. Jensen (2004). "Dependency networks for relational data." In Proceedings of the IEEE International Conference on Data Mining (ICDM), p.170-177.

R. T. Ng and J. Han (1994). "Efficient and effective clustering methods for spatial data mining." In Proceedings of the 20th Very Large Data Bases Conference, Santiago, Chile, p.144-155.

B. Padmanabhan and A. Tuzhilin (1999). "Unexpectedness as a measure of interestingness in knowledge discovery." Decision Support Systems 27: p.303--318.

L. Page, S. Brin, R. Motwani and T. Winograd (1998). "The PageRank citation ranking: bringing order to the Web." Stanford Digital Library Technologies Project.

V. Pericliev (2002). "A linguistic discovery system that verbalizes its discoveries." In Proceedings of the 19th International Conference on Computational Linguistics, p.1258-1262.

G. Piatetski-Shapiro (1991). "Discovery, analysis, and presentation of strong rules." Knowledge Discovery in Databases: p.229-248.

G. Piatetsky-Shapiro and C. J. Matheus (1994). "The interestingness of deviations." In Proceedings of the AAAI-94 Knowledge Discovery in Databases Workshop, p.25-36.

J. Pitt (1988). "Theory of Explanation." Oxford University Press, Oxford.

R. Popp, T. Armour, T. Senator and K. Numrych (2004). "Countering terrorism through information technology." Communications of the ACM 47(3): p.36-43.

J. Qin, J. Xu, D. Hu, M. Sageman and H. Chen (2005). "Analyzing terrorist networks: A case study of the global salafi Jihad Network." Lecture Notes in Computer Science.

S. Ramaswamy, R. Rastogi and S. Kyuseok (2000). "Efficient Algorithms for Mining Outliers from Large Data Sets." In Proceedings of the ACM Special Interest Group on Management of Data, p.427-438.

J. Ramon and T. Gaertner (2003). "Effcient discovery of frequent unordered trees." In Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences.

W. J. Rapaport (2002). "Holism, Conceptual-Role Semantics, and Syntactic Semantics." Minds and Machines 12(1): p.3-59.

S. Russell and P. Norvig (2003). CH 18.3: Learning decision trees. Book "CH 18.3: Learning decision trees", Prentice Hall**:** p664.

R. Schank and A. Kass (1990). "Explanations, machine learning, and creativity." Machine Learning: An Artificial Intelligence Approach 3: p.31-48.

T. Senator and H. Goldberg (2002). Break detection systems. Book "Break detection systems". W. Kloesgen and J. Zytkow, Oxford University Press**:** 863-873.

A. Silberschatz and A. Tuzhilin (1995). "On subjective measures of interestingness in knowledge discovery." In Proceedings of the First International Conference on Knowledge Discovery and Data Mining, p.275-281.

H. Simon (1995). "Machine Discovery." Foundations of Science 2: p.171-200.

B. Simons and E. H. Spafford (2003). "Letter from the Association for Computing Machinery's U.S. Public Policy Committee to the Senate Armed Services Committee."

N. Smalheiser and D. Swanson (1998). "Using Arrowsmith: a computer-assisted approach to formulating and assessing scientific hypotheses." Computer Methods and Programs in Biomedicine 57: p.149-153.

M. Sparrow (1991). "The application of network analysis to criminal intelligence: An assessment of the prospects." Social Networks 13: p.251-274.

B. Taskar, M. Wong, P. Abbeel and D. Koller (2003). "Link prediction in relational data." In Proceedings of the Neural Information Processing Systems Conference (NIPS).

R. E. Valdes-Perez (1995). "Machine discovery in chemistry: new results." Artificial Intelligence 74: p.191-201.

R. E. Valdes-Perez and V. Pericliev (1999). "Computer enumeration of significant implicational universals of kinship terminology." Cross-Cultural Research 33(2): p.162-174.

R. Vlades-Perez (1995). "Machine Discovery in Chemistry: New results." Artificial Intelligence 74: p.191-201.

J. Wang, F. Wang and D. Zeng (2006). "Rule+exception learning-based class specification and labeling in intelligence and security analysis." In Proceedings of the Workshop on Intelligence and Security Informatics, p.181-182.

S. Wasserman and K. Faust (1994). "Social Network Analysis: Methods & Applications." Cambridge, UK, Cambridge University Press.

L. Wei, W. Qian, A. Zhou, W. Jin and J. Yu (2003). "HOT: Hypergraph-Based Outlier Test for Categorical Data." In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, p.399-410.

S. White and P. Smyth (2003). "Algorithms for estimating relative importance in networks." In Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), p.266-275.

S. Wrobel (1997). "An algorithm for multi-relational discovery of subgroups." In Proceedings of the First European Symposion on Principles of Data Mining and Knowledge Discovery (PKDD-97), Berlin, p.78-87.

J. Xu and H. Chen (2005). "Criminal Network Analysis and Visualization." Communications of the ACM 48(6): p.101-107.

K. Yamanishi and J. Takeuchi (2001). "Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner." In Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), p.389-394.

Y. Yao, Y. Zhao and R. Brien (2003). "Explanation-oriented association mining using a combination of unsupervised and supervised learning algorithms." In Proceedings of the Maguire Canadian Conference on AI, p.527-531.

H. Zengyou, X. Xiaofei and D. Shengchun (2003). "Discovering cluster based local outliers." Pattern Recognition Letters 24(9-10): p.1641-1650.

T. Zhang, R. Ramakrishnan and M. Livny (1996). "Birch: An efficient data clustering method for very large databases." In Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD), Montreal, Canada, p.103-114.

N. Zhong, Y. Yao and M. Ohishima (2003). "Peculiarity oriented multidatabase mining." IEEE Transactions on Knowledge and Data Engineering 15(4): p.952- 960.

J. M. Zytkow (1996). "Incremental discovery of hidden structure: Applications in theory of elementary particles." In Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, p.750-756.

Appendices

Appendix 1: Relations and Their Descriptions in the Organized Crime Dataset

relationName(<type of source node>, <type of target node>): description
accountHolder (<BankAccount>,< Person>): the person holds the account
callerNnumber (<MakingAPhoneCall>, <Phonenumber>): The phone number of a phone-call event
ceo (<Business>,< Person>): the person is the CEO of the business
dateOfEvent(<Event>,< Date>): the date of the event, in month-date-year format
deliberateActors(<PlanningToDoSomething>,< Person>): the person plans to do something
deviceUsed (<Event>,< Device>): the device used for the event
employees (<Business>,<Person>): the person is the employee of the business
eventOccursAt (<Event>,<Place>): the event occurred at the place
geographicalSubregions (<Place1>,<Place2>): the place1 is the sub-region of place2
hasMembers (<Party>,< Person>): the party (i.e., Mafiya, business, industry) has some member
mediator (<MurderForHire>,<Person>): the person is the mediator of the contract murder event
murderer (<Event>,< Person>): the person is the murderer in the event
objectsObserved(<Observing>,<Person>): the person was observed in the observing event
operatesInRegion(<Party>,<Place>): the party (i.e., Mafiya, business, industry) operates in the place
orderedContractMurder(<MurderForHire>,< Person>): the person ordered a contract murder event
orgKiller (<Mafiya>,< person>): the person is the killer hired by the mafiya group
orgMiddleman (<Mafiya>,< Person>): the person is the middleman of the mafiya group
payee (<Paying>,< Person>): the person is the payee in the paying event
payer (<Paying>,<Person>): the person is the payer in the paying event
perpetrator (<Event>,< Person>): the person is the perpetrator in the event
phoneNumber (<Person>,< TelNumber>): the person's phone number
receiverNumber(<MakingPhoneCall>,< PhoneNumber>): the receiving number of the calling event
recipient(<Event>,< Person>): the person is the recipient in the event
relatives (<Person>,< Person>): the first person is a relative of the second

sender(<Event>,< Person>): the person is the sender in the event
socialParticipants (<Meeting>,< Person>): the person participates in the meeting
subevents(<event1>, <event2>): event1 is the subevent of event2
transferMoneyFrom (<WireTransferOfFunds>,< Bankaccount>): funds transferred from the bank account in the WireTransferOfFunds event
transferMoneyTo (<WireTransferOfFunds >,< Bankaccount>): funds transferred to the bank account in the WireTransferOfFunds event
victim(<Event>,< Person>): the person is the victim of the event
vor(<Mafiya>,< Person>): the person is the leader of the mafiya group

Appendix II: Natural Language Generation Templates for Four Explanation Strategies

2-class template with zero/non-zero separation
Node X is one of the only <integer> <node type> in the dataset (which contains a total of <integer> candidates) that: *[-<polarity> <Strings (path translation)>]** Example: uid3655 is one of the only 4 mafiyas in the dataset (which contains a total of 42 candidates) that -has a leader, and -never hired some killer, and -never has some middleman
2-class template without zero/non-zero separation
Node X is one of the only <integer> <node type> in the dataset (which contains a total of <integer> candidates) that: *[-has <more/less> than <double (0-1)> chance to be the starting node S of paths oftype <Strings>]** Example: uid1314 is one of the only 5 mafiyas in the dataset (which contains a total of 42 candidates) that -has larger than 8% chance to be the starting node S of paths of type ”S hired some killer”
3-class template with zero/non-zero separation
Node X is one of the only <integer> <node type> in the dataset (which contains a total of <integer> candidates) that: *[-<polarity> <Strings (path translation)]** *Moreover, X is different from the other <integer> nodes because it* *[-<polarity> <Strings (path translation), while others do/don’t]** Example: uid3655 is one of the only 4 mafiyas in the dataset (which contains a total of 42 candidates) that has some member paying some money to somebody, and never hired some killer, and has some member receiving some money from somebody Moreover, uid3655 is different from the other 3 mafiyas because it has a leader who is the recipient of some communication event, while others don’t never has some middleman, while others do

3-class template without zero/non-zero separation
Node X is one of the only <integer> <node type> in the dataset (which contains a total of <integer> candidates) that: [-has <more/less> than <double (0-1)> chance to be the starting node S of paths of type <Strings>}* Moreover, X is different from the other <integer> nodes because it [-has <polarity> <<double> v.s. <double> > chance to be the starting node S of paths of type <Strings>]* Example: uid2241 is one of the only 8 mafiyas in the dataset (which contains a total of 42 candidates) that -has smaller than 1.042% chance to be the starting node S of paths of type “S has some member participating in some social event”, and -has smaller than 0.01% chance to be the starting node S of paths of type ”S hired some killer who has phone number recorded”, and -has larger than 2.21% chance to be the starting node S of paths of type “S has some member who is also a member of other mafiya group” Moreover, uid2241 is different from the other 7 node(s) because it -has relatively higher (10% v.s 0%) chance to be the starting node S of paths of type “S has some member who is a perpetrator”

Appendix III: 2-class explanations provided for human subjects for Task 2

(The crime participants are Uid4542, Uid4502, Uid3655)

Uid707 is one of the only 4 mafiyas in the dataset (which contains a total of 42 candidates) that –never has some member who is the sender of a communication event –has some member planning to do something –never hired some killer
Uid3655 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has a leader who paid some money to somebody Also , Uid3655 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has a leader who is the recipient of a communication event –has a leader who is the member of some other mafiya
Uid3695 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –never has some member participating in some meeting –has some member paying some money to somebody –never hired some killer who paid some money to somebody
Uid4582 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has a middleman who is a member of some mafiya –never has some middleman participating in some meeting –has some member receiving some money from somebody
Uid4542 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has some member who is the victim for a murder event Also, Uid4542 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has some member who observed a murder event Also , Uid4542 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has a leader who is also the leader for other mafiya group
Uid1474 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has some middleman participating in some meeting –never has some member who is the sender of a communication event –never has some middleman also working as a middleman for other mafiya
Uid4502 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has a member who ordered a contract murder Also , Uid4502 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has some middleman planning to do something –never has some middleman who is the sender of a communication event
Uid2121 is one of the only 3 mafiyas in the dataset (which contains a total of 42 candidates) that –never has some member who is the recipient of a communication event –never hired some killer –has some member planning to do something
Uid2888 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that –has a leader who is the sender of a communication event –never has a leader who participated in some meeting Also , Uid2888 is one of the only 12 mafiyas in the dataset (which contains a total of 42 candidates) that –never has some member participating in some meeting

Uid5349 is the only 1 mafiya in the dataset (which contains a total of 42 candidates) that
–has some middleman participating in some meeting
–never has some member participating in some meeting
Also , Uid5349 is one of the only 10 mafiyas in the dataset (which contains a total of 42 candidates) that
–has some middleman who is the recipient of a communication event

3-class explanations provided for human subjects for Task 3

Uid3695 is one of the only 4 mafiyas in the dataset (which contains a total of 42 candidates) that
– has some member paying some money to somebody
– never has some member who is the sender of a communication event
– has some member planning to do something
Moreover, Uid3695 is different from the other 3 mafiyas because it
– has a leader who has some phone number recorded, while others didn't
– has a leader who has some bank account recorded, while others didn't
– never has some member who is the recipient of a communication event, while others did

Uid1474 is one of the only 4 mafiyas in the dataset (which contains a total of 42 candidates) that
–never has some member who is the sender of a communication event
–has some member who received some information from somebody
–never has some member participating in some meeting including a information sender
Moreover, Uid1474 is different from the other 3 mafiyas because it
–has some middleman who has phone number recorded, while others didn't
–has some middleman who has some bank account recorded, while others didn't

Uid5349 is one of the only 3 mafiyas in the dataset (which contains a total of 42 candidates) that
–never has some member participating in some meeting
–never hired some killer
–never has a leader
Moreover, Uid5349 is different from the other 2 mafiyas because it
–has some member receiving some money from somebody, while others didn't
–has some middleman who is the recipient of a communication event, while others didn't
–has some middleman participating in some meeting, while others didn't

Uid4582 is one of the only 5 mafiyas in the dataset (which contains a total of 42 candidates) that
–has some member receiving some money from somebody
–never hired some killer
–never has some middleman participating in some meeting
Moreover, Uid4582 is different from the other 4 mafiyas because it
–has some middleman who has phone number recorded, while others didn't
–has some middleman who has some bank account recorded, while others didn't
–has a member who is the middleman of some mafiya, while others didn't

Uid2888 is one of the only 3 mafiyas in the dataset (which contains a total of 42 candidates) that
–never has some member participating in some meeting
–never hired some killer
–never has some middleman
Moreover, Uid2888 is different from the other 2 mafiyas because it
–has some member receiving some money from somebody, while others didn't

Uid707 is one of the only 4 mafiyas in the dataset (which contains a total of 42 candidates) that
–never has some member who is the sender of a communication event
–has some member who received some information from somebody
–never has some member receiving some money from somebody
Moreover, Uid707 is different from the other 3 mafiyas because it
–never has some member whose phone number is the receiving number of a communication event, while others did

Uid2121 is one of the only 8 mafiyas in the dataset (which contains a total of 42 candidates) that
–never has some member receiving some money from somebody
–never hired some killer
–never has some middleman
Moreover, Uid2121 is different from the other 7 mafiyas because it
–has some member who sent some message to some people, while others didn't

Uid4502 is one of the only 10 mafiyas in the dataset (which contains a total of 42 candidates) that
–never has some member receiving some money from somebody
–never hired some killer
–never has some member who is a perpetrator
Moreover, Uid4502 is different from the other 9 mafiyas because it
–has some middleman participating in some meeting, while others didn't
–has a member who ordered a contract murder, while others didn't

Uid4542 is one of the only 2 mafiyas in the dataset (which contains a total of 42 candidates) that
–has a leader who is the sender of a communication event
–never has a leader who is the member of some other mafiya
–never has a leader who is the perpetrator
Moreover, Uid4542 is different from the other 1 mafiya because it
–has some member who is the sender of a communication event, while the other one didn't
–has some member participating in some meeting, while the other one didn't
–has some member planning to do something, while the other one didn't

Uid3655 is one of the only 4 mafiyas in the dataset (which contains a total of 42 candidates) that
–has some member paying some money to somebody
–never hired some killer
–has some member receiving some money from somebody
Moreover, Uid3655 is different from the other 3 mafiyas because it
–has a leader who is the recipient of a communication event, while others didn't
–has a leader who paid some money to somebody, while others didn't
–never has some middleman, while others did