

Link Analysis of Higher-Order Paths in Supervised Learning Datasets

Murat C. Ganiz, William M. Pottenger and Xiaoning Yang
Lehigh University
{mug3, billp, xiy204}@lehigh.edu

Keywords: Link Analysis, Link Mining, Higher-Order Links/Paths, Supervised Learning, Decision Trees

Abstract

Due to recent concerns with security and terrorism there has been an increasing focus on techniques that discover links and relations in data. Several efforts that employ “data mining” techniques have contributed to this field, but few focus on discovering patterns in sets of higher-order links, which can reveal hidden or indirect relationships in data. In this work we focus on the discovery and analysis of higher-order path patterns in a supervised learning dataset. We first analyze higher-order links in the leaf nodes of a decision tree and find evidence for distinguishing between nodes of different classes. Based on these results we next focus on the training data itself used to build the tree. Our results indicate that classes of instances in labeled training data may be separable based on the characteristics of higher-order paths.

Introduction

Due to recent concerns with terrorism and security there has been an increasing focus on techniques that discover links and relations in data. Numerous efforts that employ “data mining” techniques have contributed to this field, and of these several focus on higher-order links, which can reveal hidden or indirect relationships in data. Of these, however, few if any have studied the patterns or characteristics in sets of higher-order links to discover meaningful relationships.

To give a proper context for our work it is useful to consider efforts that employ higher-order link analysis in security, counterterrorism and law enforcement applications. One example is given in [18], in which algorithms for discovering shortest paths are used to assist in fighting organized crime through identifying associations in criminal networks. This work first applies named entity extraction [5] to extract entities from unstructured textual data. It then constructs a semantic network [4], a technique used in COPLINK® Detect [9]. As noted, the technique then employs a shortest-path algorithm to discover important relations between given entities. This approach has been tested on Phoenix Police department crime reports, and results in [18] indicate that the relations mined are useful about 70% of the time.

A related effort that also provides context for our own work employs a Link Discovery algorithm in a counterterrorism application [21]. This work is part of the DARPA Evidence Extraction and Link Discovery (EELD) program, and aims to discover

links (patterns) from multi-relational data. This approach employs an Inductive Logic Programming (ILP) covering algorithm and a labeled training dataset to discover significant links. The solution has been tested on nuclear smuggling and contract killing data sets and shown in [21] to have better performance than a baseline.

In this paper, we focus on discovering higher-order link patterns in data based on a co-occurrence relationship between entities. In this context, a higher-order link can be represented as a chain of co-occurrences of entities in different records. We also refer to such a link as a higher-order path. Given a supervised learning dataset (i.e., labeled training data), we attempt to discover patterns in sets of higher-order links that distinguish between the classes in the labeled data.

The work in [18] has some similarity with ours. Both employ co-occurrence as the relationship between entities and concentrate on the higher order co-occurrence relations or paths. The order of the relation (i.e., the length of the path) ranges from second order (e.g., as in [14]) on up. The approach in [18], however, is focused on discovering significant paths between entities such as a link between two terrorism suspects. In contrast, our approach focuses on discovering patterns in sets of the higher-order links themselves. In other words, we study the characteristics of sets of higher-order paths with the overall goal of performing classification of labeled instances based on the characteristics of these higher-order path sets.

As noted the effort discussed in [21] employs a supervised machine learning algorithm and labeled training data. Our work is similar in that we also employ labeled training data. The goal in [21] is to learn higher-order link rules; i.e., rules that are themselves higher-order links between sets of entities. In contrast, as noted our goal is to discover the characteristics of sets of higher-order paths with the goal of leveraging these characteristics in classification. Naturally, such a technique would have wide application in supervised machine learning, including the link analysis research field. Our preliminary results are based on a labeled training dataset from the UCI repository [12], but our technique can be extended for use with various supervised learning datasets in security, law enforcement and counterterrorism applications.

Related Work

In this article we often refer to the terms

“higher-order path,” “higher-order link” or “higher-order co-occurrence.” In order to understand the meaning of these terms, consider figure 1 below (reproduced from [11]). This figure depicts three documents, D1, D2 and D3, each containing two terms, or entities, represented by the letters A, B, C and D. Below the three documents is a higher-order path that links entity A with entity D through B and C. This is a third-order path since three links, or “hops,” connect A and D.

In what follows D1, D2 and D3 are not always documents – they might be records in a database or instances in a labeled training dataset. Likewise, the entities A, B, C etc. need not be terms – they may be values in a database record, or items (attribute-value pairs) in an instance.

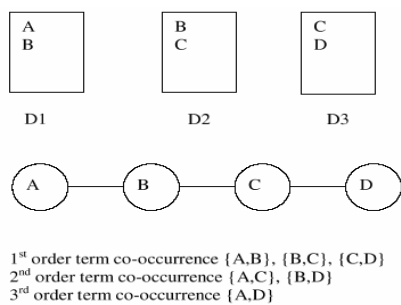


Figure 1: Higher-order co-occurrences [11]

Higher-order co-occurrences play a key role in the effectiveness of systems used for information retrieval and text mining. One example is Literature Based Discovery (LBD), which employs second-order co-occurrence to discover connections between concepts (entities). A well-known example is the discovery of a novel migraine-magnesium connection in the medical domain [14]. The authors of [14] found that in the Medline database some terms co-occur frequently with “migraine” in article titles, e.g. “stress” and “calcium channel blockers.” They also discovered that “stress” co-occurs frequently with “magnesium” in other titles. As a result, they hypothesized a link between “migraine” and “magnesium,” and some clinical evidence has been obtained that supports this hypothesis. In LBD a second-order link of this nature is represented as $A \rightarrow B \rightarrow C$ where in this example A is “migraine,” C is “magnesium” and B is one of several possible connecting terms such as “stress.”

In our previous work in [11], we proved mathematically that Latent Semantic Indexing (LSI), a well-known approach to information retrieval, implicitly depends on higher-order co-occurrences. We also demonstrated empirically that higher-order co-occurrences play a key role in the effectiveness of systems based on LSI.

LSI can reveal hidden or latent relationships among terms, as terms semantically similar lie closer to each other in the LSI vector space. This

can be demonstrated using the LSI term-term co-occurrence matrix. Let’s assume a simple document collection where D1 is {human, interface} and D2 is {interface, user}. Clearly the terms “human” and “user” do not co-occur in the co-occurrence matrix of this simple two-document collection. After applying LSI, however, the reduced representation co-occurrence matrix may have a non-zero entry for “human” and “user” thus implying a similarity between the two terms. This is an example of second-order co-occurrence; in other words, there is a second-order path between “human” and “user” through “interface.”

The results of experiments reported in [11] show that there is a strong correlation between second-order term co-occurrence, the values produced by the Singular Value Decomposition (SVD) algorithm used in LSI, and the performance of LSI measured in terms of F_β , the harmonic mean of precision and recall. As noted, [11] also provides a mathematical analysis which proves that LSI in fact depends on higher-order term co-occurrence.

The results of this prior work persuade us that higher-order paths may be important in other domains as well, such as supervised machine learning. We discuss this in the Approach section below. Before exploring this possibility further, however, it is worth noting that higher-order co-occurrence plays an important role in many other systems used for information retrieval and text mining. In what follows we describe a few applications that either explicitly or implicitly use higher order co-occurrence.

In [7], Edmonds uses co-occurrence to solve a component of the problem of lexical choice, which identifies synonyms in a given context. The approach uses second- or higher-order co-occurrence to predict the most likely synonym. Relations between synonyms are represented in a lexical co-occurrence network such as that depicted in figure 2. Starting with a given root term, significant co-occurring terms are added to the tree as children of the root term. This process continues recursively, each level of the tree increasing the order of co-occurrence with the root term.

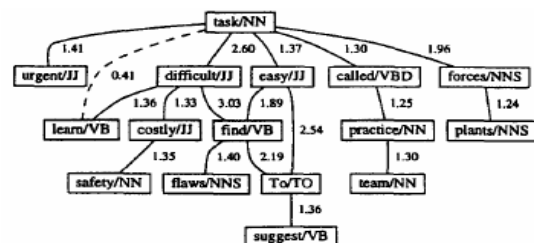


Figure 2: A fragment of a lexical co-occurrence network. The dashed line is an example of a second-order path implied by the network [7]

Zhang et al. [20] use second-order co-occurrence to improve the runtime performance of LSI. Although LSI has been shown to perform

well in terms of precision and recall, due to the high execution time of SVD, the authors apply information filtering to improve the runtime performance. In [20], they narrow the term and document space, starting from the query, by following second-order paths between the query and the collection. The resulting set of documents is then expanded by repeating the process and following second-order paths between the selected documents and the remainder of the collection. As a result, a much smaller subset of the original collection is extracted for each query, and the resulting term by document matrix contains only about 27% of the original non-zero entries on average. Based on this approach the new matrix achieved 65% reduction of the original non-zero entries on average, and only a 5% loss in precision for most collections.

Second- and higher-order co-occurrence has also been used in a number of other applications including word sense disambiguation [13] and in a stemming algorithm [19].

One of the challenges facing us in this work is the complexity of enumerating the various higher-order paths. In this area too, fortunately, there has been prior work on which we can build. In the link mining domain, for example, Chakrabarti et al. [3] use the notion of a directed bipartite graph to discover communities in such graphs that have a large density of edges. They predict that many of these graphs will contain smaller bipartite complete subgraphs: e.g., each node in subgraph A has a link to each node in subgraph B. Using a variety of pruning algorithms they enumerate all such complete bipartite matchings. In a related effort, Sun et al. [22] use a bipartite representation of several different applications including P2P systems and research publications. They introduce two operations on bipartite graphs; first, the identification of similar nodes used for relevance search, and second, the discovery of edges connecting irrelevant nodes for anomaly detection.

In [8] Galil surveys techniques used for designing efficient algorithms for finding a maximum cardinality or weighted matching in (general or bipartite) graphs. For a bipartite graph $G = (V, E)$, perfect matchings are defined as matchings such that all vertices are incident to some matching edge. On the other hand, maximum matchings are defined as matchings whose cardinalities are maximum among all matchings, and maximal matchings are matchings which are contained in no other matching. Uno [15] also presents enumerating algorithms for perfect, maximum and maximal matchings in bipartite graphs. An algorithm that has a time complexity of $O(n)$ per matching is proposed for maximal matchings in bipartite graphs.

Another way to view the challenge of enumerating higher-order paths is to view the paths of interest as a system of distinct representatives (SDRs). An SDR of the sets A_0, \dots, A_{n-1} is defined as a sequence of n distinct elements a_0, \dots, a_{n-1} with $a_i \in A_i, 0 \leq i \leq n-1$ [16]. A system of distinct representatives is equivalent to a maximal matching on some bipartite graph [1].

These efforts serve to highlight the fact that the enumeration of higher-order paths can be approached from at least two different perspectives, and as we show in the Approach section following, it is also possible to leverage the inclusion/exclusion principle [16]. In addition, although enumeration of higher-order paths is in general number-complete (i.e., p-sharp), this related work reveals that less complex algorithms exist for partial enumeration.

Approach

Our definition of a higher-order path is similar to that found in graph theory, which states that given a non-empty graph $G = (V, E)$ of the form $V = \{x_0, x_1, \dots, x_k\}$, $E = \{x_0x_1, x_1x_2, \dots, x_{k-1}x_k\}$ with nodes x_i distinct, two vertices x_i and x_k are linked by a path P where the number of edges in P is its length. Such a path is often referred to by the natural sequence of its vertices $x_0x_1\dots x_k$. [6]. Our definition of a higher-order path differs from this in a couple of respects. First, vertices $V = \{e_0, e_1, \dots, e_k\}$ represent entities, and edges $E = \{r_0, r_1, \dots, r_m\}$ represents records, documents or instances. Several edges may exist between given entities. Finally and most importantly, in a higher-order path both vertices and edges must be distinct. Figure 3 gives an example of several higher-order paths such as $e_1-r_1-e_2$, $e_2-r_2-e_3-r_3-e_4$, $e_2-r_6-e_5$, etc. We are interested in enumerating all such paths.

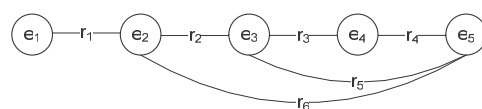


Figure 3: An example of a fourth-order path between e_1 and e_5 , as well as several shorter paths

Co-occurrence relations in a record or instance set can be represented as an undirected graph $G = (V, E)$ such that V is a finite set of vertices (i.e., entities) and E is the set of edges representing co-occurrence relations. In other words, if two entities co-occur in a record then there is an edge between the corresponding vertices and this edge is labeled with the records(s) in which they co-occur. It is not straightforward, however, to depict higher-order paths with conventional graph structures because multiple paths may connect two given entities: for example, $e_1-r_1-e_2$ and $e_1-r_2-e_2$ are both valid paths between entities e_1 and e_2 . A conventional graph can nonetheless be modified to

represent paths of this nature by maintaining a data structure that contains lists of records for each edge. We term this a *path group*. Path groups are extracted directly from the co-occurrence graph G . Using this representation, the higher-order paths correspond to a complete matching in the bipartite graph formed from the set of entities and the set of lists of records. Likewise, higher-order paths defined in this manner are the system of distinct representatives of the sets of records for each edge.

Both theory and experiments reveal that even with a small dataset of a couple hundred records each containing less than ten entities, the number of higher-order paths can be very large. This naturally raises severe time and space complexity issues when enumerating higher-order paths. To demonstrate this point, assume that we have 100 records each containing seven entities in a dataset; for simplicity we also assume that each record is identical. For an n^{th} -order path, we need to select $n+1$ entities out of the seven available (combinations) and these can be arranged in $(n+1)!$ ways (permutations). This leads to the following formula enumerating all of the higher-order paths in this dataset:

$$C(7, n+1) * (n+1)! * \prod_{i=0}^{n-1} (100 - i)$$

In all, there are 2,079,000 second-order paths, 814,968,000 third-order paths, 237,155,688,000 fourth-order paths, and so on up to sixth-order. In order to address these complexity issues, we must limit the size of the datasets explored. As our results in the next section indicate, however, we nonetheless have discovered interesting patterns in the higher-order path data.

As stated in the introduction, our goal is to characterize the set of higher-order paths – in other words, we are seeking patterns in the higher-level path data itself. As a result, we need to enumerate and possibly store all the paths in a given dataset. This required the development of special data structures. We based our implementation on the Text Mining Infrastructure (TMI) developed by the Parallel and Distributed Text Mining Lab at Lehigh University [10]. The TMI is an open-source framework designed for high-end, scalable text mining, and aims to provide a robust software core for research and development of text mining applications. The TMI has an inverted index class that provides an easy and efficient way to extract co-occurrence relations between entities.

In our first approach to the problem we represented higher-order paths as chains of co-occurrences such as that depicted in figure 4. Based on this representation we developed a recursive algorithm that takes two distinct entities and a stop level as a parameter and enumerates all

the higher-order paths up to the stop level. Due to the need to store paths on disk efficiently for post-processing, however, we modified the data structure to directly represent path groups. This provided a reasonably efficient framework within which to conduct experiments.

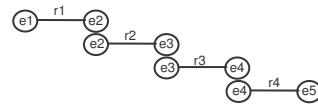


Figure 4: Fourth-order path as a chain of first-order co-occurrences

Once a path group is formed for a given order, the number of paths in a given group must be enumerated. As noted this is a complex computation, which led us to explore the possibility of representing this count in a closed form. To investigate this possibility, we leveraged the well-known inclusion-exclusion principle.

Given that A and B are subsets of S and one wishes to count the elements of $S \setminus (A \cup B)$, then the idea of inclusion-exclusion is to compute $|S| - |A| - |B| + |A \cap B|$ [16]. (Interestingly, the sum is not the more straightforward $|S| - |A| - |B|$ because the elements of $A \cap B$ would be subtracted twice.)

Based on this principle we developed the following closed-forms for enumerating second- and third-order paths in path groups:

$$\text{Second-order: } |A| * |B| - |A \cap B|$$

$$\text{Third-order: } |A| * |B| * |C| - (|A \cap B| * |C| + |A \cap C| * |B| + |B \cap C| * |A|) + 2 * |A \cap B \cap C|$$

Here the letters A , B and C represent sets of records for each edge in a third-order path group (e.g. $e_1 - \{2,3,4\} - e_2 - \{1,3,5\} - e_3 - \{2,3,5\} - e_4$, $A = \{2,3,4\}$, $B = \{1,3,5\}$, $C = \{2,3,5\}$) Although our application is not completely analogous to the enumeration of sets using the inclusion-exclusion principle, we were able to successfully develop these two closed-forms using inclusion-exclusion as a starting point. However, when enumerating fourth- and higher order paths from the path groups, our extension of the inclusion-exclusion principle cannot be applied in the same way. As a result, we have taken a different tack and are developing closed-forms for fourth- and higher-order paths based on both SDR and bipartite graph representations. This work is ongoing, and at the time of writing we are close to discovering the closed-form for fourth-order paths.

Since our goal is to discover patterns in the higher-order paths, we first employed an association rule mining algorithm to discover rules correlating entities in higher-order paths. We modified the code developed by Borgelt and Kruse [2] to read path groups from a file and extract all

valid paths. Each path is considered a transaction and the output is frequent itemsets and rules. Due to the very large number of higher-order paths in our datasets, however, the algorithm was unable to compute the rules on a 32-bit architecture. As a result, we implemented our own method to discover frequent itemsets in the higher-order paths. However, our definition of frequent itemsets is a bit different from the standard definition used in association rule mining. Itemsets in our framework are ordered, and must appear in order in a given supporting path (transaction). Additionally, the items (entities) in an itemset must be adjacent in the higher-order path.

During computational enumeration of the paths, statistics are gathered. Specifically, in order to characterize a given set of records/instances, we compute frequencies of the various second- and higher-order itemsets in the set of all higher-order paths generated from the set of instances. When dealing with labeled training data used in supervised machine learning, we divide the instances by class and then characterize the resulting sets by higher-order itemset frequency. In the case of decision tree models, we take one or more nodes of each class, group nodes of the same class together, and characterize the resulting groups. The end result is a distribution of itemset frequencies for a given class. These distributions can be compared using simple statistical measures such as a t-test to determine independence. If two distributions are statistically significantly different, we conclude that the higher-order path patterns (i.e., itemset frequencies) distinguish the classes.

Results, Analysis and Discussion

As noted previously, the implementation of our algorithm is based on the TMI [10] and thus implemented in C++. We performed the experiments to discover the higher-order path statistics on the National Center for Supercomputing Applications (NCSA) Tungsten Supercluster (Xeon Linux). Tungsten is composed of Intel 3.06GHz Xeon DP processor-based systems running Red Hat 9.0, with Myrinet 2000 interconnects, an I/O subcluster with more than 120 terabytes of DataDirect storage. Tungsten provides Intel 8.0 icc and GNU gcc 3.2.2 for compilation, the Load Sharing Facility (LSF) batch system for job control and ChaMPIon/Pro for the MPI runtime environment. The GNU C++ compiler was used with aggressive optimization parameters.

The supervised machine learning dataset employed was drawn from the UCI machine learning repository. This is a repository of training sets, domain theories and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms [12]. The dataset includes mushroom records drawn from The Audubon Society Field Guide to

North American Mushrooms, described in terms of physical characteristics. This data contains descriptions of samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as edible (E class) or poisonous (P class). The dataset contains 8,124 instances and 22 nominal attributes. The class distribution is 4208 edible (51.8%) and 3916 poisonous (48.2%). It is a relatively large training set that performs quite well on a standard C4.5-based decision tree induction algorithm. In addition, we selected this dataset for our initial experiments because all the attributes are nominal, making it straightforward to use the TMI to represent the higher-order paths. The choice to employ the TMI also opens the way to explore patterns in higher-order paths in textual data sources in the future. Finally, this dataset has only two different classes, simplifying the process of grouping instances by class and increasing the representation of each class.

As noted in the Approach, we initially applied a decision tree algorithm to the dataset. We employed the J4.8 classifier in the WEKA Workbench, a collection of machine learning algorithms [17]. Since the TMI has the ability to invoke a WEKA classifier via the Java Native Interface (JNI), we designed our code to read instances from the leaf nodes of the decision tree learned from the mushroom training data. As noted, our goal was to see whether the characteristics of the higher-order paths could distinguish between E (edible) and P (poisonous) leaf nodes.

Our preliminary results indicated that the number of higher-order paths may be correlated with the number of unique attribute-value pairs (AVPs) in the dataset. Thus, due to the computational complexity of the enumeration, we selectively removed attributes which had many values. For example, the cap-color and gill-color attributes had 10 and 12 distinct values respectively and were removed. Other attributes with two or three distinct values were retained. Proceeding in this way we reduced the number of attributes from 22 to seven, including the class attribute (E or P). Despite this quite significant reduction in the dataset, the 10-fold cross-validation performance in WEKA was still about 93%. The resulting decision tree had 27 nodes of which 17 were leaf nodes. This reduction in the number of attributes drastically lessened the running time of our path enumeration algorithm.

As an example of the enumeration results, one of the E class leaf nodes with 96 instances contained 1,094,400 second-order paths and 308,620,800 third-order paths. Similarly, another pure E leaf node with 192 instances had 4,400,640 second-order and 2,508,364,800 third-order paths. As can be seen from this example, when the number of instances doubles, the number of

second-order paths increases about four times and third-order paths increase about eight times. Thus the number of instances in a set also has a direct impact on the number of higher-order paths.

Due to the computational complexity of the enumeration, we selected three small leaf nodes from each class and analyzed the higher-order paths by class. All of these nodes were pure, and together yielded a set of 304 E instances composed of 12 unique AVPs and 292 P instances composed of 10 AVPs. Table 2 depicts the outcome of these experiments. In tabulating the results in table 1, we disallowed first-order co-occurrence between the first and last items in a given higher-order path. It is worth noting that although the P instances contain only 10 unique AVPs, the second- and third-order paths in table 2 are more numerous than those discovered in the E instances.

Table 1: Path counts with restriction applied

Paths	E-leaf paths	P-leaf paths
2nd order	912,384	491,688
3rd order	465,315,840	154,773,504
4th order	225,364,561,920	42,780,856,320

Table 2: Path counts without restriction

Paths	E-leaf paths	P-leaf paths
2nd order	7,062,912	9,053,352
3rd order	1,269,611,136	1,555,160,112
4th order	285,724,593,216	282,781,502,748

Enumerating the higher-order paths is actually just a step towards our goal of characterizing the paths. To gather statistics from the path groups discussed earlier, we created a list of 3-itemsets from the fourth-order paths ranked by their frequencies. This frequency is similar to the support statistic used in association rule mining. For a given 3-itemset, the frequency is the number of fourth-order paths that the itemset occurs in. Each fourth-order path contains three such 3-itemsets. In the process of tabulating these statistics, we discovered that several itemsets have the same frequency. Thus, we formed a ranked list of itemset frequencies as well as a count of itemsets of each frequency. A count greater than one indicates more than one itemset occurred with the same frequency. As noted in the Approach, we then applied a simple statistical test, the t-test, to the lists of 3-itemsets same-frequency counts for the E and P datasets respectively using paths without restriction (table 2). We found that these two distributions are statistically significantly different with a confidence greater than 95%: the results of the t-test are given in table 3 below.

These results indicate that we may be able to distinguish between classes in leaf nodes of decision trees based on the characteristics of the higher-order paths. A similar result was obtained

for the counts of frequencies of 3-itemsets from third-order paths, each of which contain two 3-itemsets. Again, this time with greater than 99% confidence, the E-leaves and P-leaves from our test dataset could be separated based only on the characteristics of the higher-order paths (table 4).

Table 3: Two-sample t-test applied to the counts of 3-itemset frequencies from fourth-order paths

	E-leaves	P-leaves
Mean	8.357142857	6.491228
Variance	33.68831169	11.682957
Observations	56	57
Hypothesized	0	
Degrees freedom	89	
t Stat	2.077679424	
P(T<=t) one-tail	0.020309856	
t Critical one-tail	1.662155326	
P(T<=t) two-tail	0.040619712	
t Critical two-tail	1.986978657	

Table 4: Two-sample t-test applied to the counts of 3-itemset frequencies from third-order paths

	E-leaves	P-leaves
Mean	15.09677419	8.409090909
Variance	107.6903226	34.85200846
Observations	31	44
Hypothesized Diff	0	
Degrees freedom	44	
t Stat	3.237924166	
P(T<=t) one-tail	0.001146178	
t Critical one-tail	1.680229977	
P(T<=t) two-tail	0.002292356	
t Critical two-tail	2.015367547	

Based on these results we extended our scope to obtain statistics not only from decision tree leaf nodes but directly from the labeled training data itself. Since our experimental dataset has a large number of instances, we divided it into ten folds. To accomplish this, the dataset first had to be randomized using an instance randomization filter. Following this, we randomly selected instances from each class to form folds. Each resulting fold had 391 E and 391 P instances. We created folds of the same size in order to mitigate the effect of the number of instances on the number of higher-order paths. We enumerated the number of higher-order paths for each fold for each class. The results are depicted in tables 5, 6 and 7. Following this we gathered statistics on the frequency of 3-itemsets in fourth-order paths with no restriction on the paths. Finally, we counted the number of same-frequency itemsets for each unique frequency.

Table 5: Second-order paths in E and P instances

Fold	E-instances	P-instances
0	13374796	14963992
1	13802640	14807288
2	14149972	15128396
3	14140364	15014408
4	13658568	14806900
5	13623148	14909464
6	14027660	14962836
7	13609504	14896052
8	13786768	14837246
9	13756428	14806690

Table 6: Third-order paths in E and P instances

Fold	E-instances	P-instances
0	12376281978	14921747380
1	13087031162	14669936660
2	13644072116	15211709372
3	13615318426	15028870504
4	12808481968	14656258974
5	12794667846	14830395830
6	13441211064	14921668250
7	12760275832	14822502030
8	13058138066	14723983262
9	13005820302	14648564880

Table 7: Fourth-order paths in E and P instances

Fold	E-instances	P-instances
0	8.90419E+12	1.11037E+13
1	9.47334E+12	1.08798E+13
2	9.94377E+12	1.13544E+13
3	9.92627E+12	1.11849E+13
4	9.27069E+12	1.08781E+13
5	9.23824E+12	1.10276E+13
6	9.777E+12	1.11057E+13
7	9.21041E+12	1.10071E+13
8	9.45475E+12	1.09242E+13
9	9.40945E+12	1.08777E+13

Table 8: Unique AVPs in E and P instances

Fold	AVPs in E	AVPs in P
0	12	13
1	12	13
2	12	13
3	12	13
4	12	13
5	12	13
6	12	13
7	12	13
8	12	13
9	12	12

As can be seen from these results, there is a discernable pattern in the higher-order path frequencies tabulated for each class – there are consistently more paths in the P-instances. However, as noted previously there are factors that influence the number of higher-order paths: the

number of instances, and the number of AVPs. We ruled out the number of instances as a factor by creating folds with equal numbers of E and P instances. In our empirical studies, however, we found that the number of distinct AVPs in a dataset may also impact the number of higher-order paths. Thus, in order to evaluate the potential impact of differing numbers of AVPs in P and E sets, we tabulated (table 8) and analyzed them.

We observe that there is again a different pattern between the E and P classes. In all folds except one, there are more AVPs in the P instances. Since each dataset has the same number of instances, the difference in the number of higher-order paths might be due to the number of AVPs. In fold 9, however, both the E and P instances have the same number of instances and AVPs. Yet the pattern in the number of paths holds for fold nine. I.e., the number of higher-order paths is greater for the P-instances in fold nine, continuing the trend found in the other folds. The fact that the trend holds encouraged us to proceed to analyze the 3-itemset frequencies in the higher-order paths as we had done before.

Unlike the previous experiment with the decision tree leaf nodes, however, in this experiment (with the E and P instances drawn directly from the labeled training data) we found that the counts of same-frequency itemsets yielded no significant information. In other words, we were unable to distinguish between the E and P classes using the counts of same-frequency itemsets. As a result, again using the t-test we compared the E and P distributions using the 3-itemset frequencies themselves. The results of these comparisons are depicted in table 9.

Table 9: Two-sample t-test assuming unequal variances between E and P instances

Fold	t Stat	P(T<=t) one-tail	t _{Critical} one-tail	P(T<=t) two-tail	t _{Critical} two-tail
0	-2.684	0.0037	1.6471	0.0074	1.9634
1	-1.357	0.0875	1.6467	0.1751	1.9629
2	-1.554	0.0603	1.6468	0.1205	1.9629
3	-2.924	0.0018	1.6472	0.0036	1.9636
4	-1.908	0.0284	1.6469	0.0568	1.9631
5	-2.047	0.0205	1.6469	0.041	1.9631
6	-1.455	0.073	1.6467	0.146	1.9629
7	-2.023	0.0217	1.6469	0.0434	1.9631
8	-2.795	0.0027	1.6471	0.0053	1.9635
9	-2.71	0.0034	1.647	0.0069	1.9633

As can be seen from the table 9, six of the 10 folds had a confidence of 95% or greater that the E

and P instances are significantly statistically different. In order to ascertain whether this pattern also holds when comparing like instances, we also performed both E-to-E and P-to-P statistical tests. These results are depicted in tables 10 and 11, and show that in no case was there a statistically significant difference between two folds when comparing instances of the same class. Thus we again draw the tentative conclusion that the frequency distributions of itemsets of higher-order paths may capture distinguishing characteristics of the classes in supervised machine learning training datasets. Although these results are preliminary in nature and hold only for the experimental dataset we evaluated, as noted they indicate that classes of instances in labeled training data may be separable using the characteristics of higher-order paths.

Table 10: Two-sample t-test assuming unequal variances between E and E instances

Folds	t Stat	P(T<=t) one-tail	t Critical one-tail	P(T<=t) two-tail	t Critical two-tail
0-1	-0.627	0.2654	1.6467	0.5308	1.963
0-2	-1.083	0.1395	1.6467	0.279	1.963
0-3	-1.074	0.1416	1.6467	0.2832	1.963
0-4	-0.425	0.3353	1.6466	0.6706	1.963
0-5	-0.382	0.3515	1.6467	0.7029	1.962
0-6	-0.931	0.176	1.6467	0.352	1.963
0-7	-0.351	0.3627	1.6467	0.7254	1.963
0-8	-0.61	0.2709	1.6467	0.5417	1.963
0-9	-0.563	0.2867	1.6467	0.5733	1.963

Why do patterns in higher-order paths seem to correlate with the class? In a sense it hearkens back to our prior work with Latent Semantic Indexing (LSI) [11] – in that work, as noted, we determined that the ‘Latent’ aspects of term similarity that LSI reveals are dependent on the higher-order paths between terms. Likewise, in real-world supervised machine learning datasets, the goal is to learn the relation between the attributes and the class. It is noteworthy that attributes are certainly not equally important. In addition, neither attributes nor instances are independent of one another, given the class. As we found with LSI, it is our contention that the ‘latent semantics’, if you will, of attribute-attribute relations also depend on the higher-order paths linking attribute-value pairs. By taking attribute-value pairs as our base unit of ‘semantics’ and linking them via higher-order co-occurrence relations, we reveal these latent semantics, or patterns, that distinguish instances of different classes. These preliminary results are extremely interesting given that we have uncovered evidence of separability *without the use of a*

supervised machine learning algorithm! We consider this achievement significant, and something that can be exploited in many different applications using a variety of datasets as long as there is a meaningful context of entities that allows us to leverage co-occurrence relations. In the following section we discuss some potential applications of this work.

Table 11: Two-sample t-test assuming unequal variances between P and P instances

Folds	t Stat	P(T<=t) one-tail	t Critical one-tail	P(T<=t) two-tail	t Critical two-tail
0-1	0.6937	0.244	1.6467	0.4881	1.9629
0-2	-0.002	0.4991	1.6467	0.9983	1.9629
0-3	-1.349	0.0889	1.647	0.1778	1.9633
0-4	0.4156	0.3389	1.6467	0.6778	1.9629
0-5	0.2655	0.3953	1.6467	0.7907	1.9629
0-6	0.2879	0.3867	1.6467	0.7735	1.9629
0-7	0.3029	0.381	1.6467	0.762	1.9629
0-8	-0.704	0.2408	1.6469	0.4816	1.9631
0-9	-0.532	0.2974	1.6468	0.5948	1.963

Conclusions and Future Work

Due to the recent concerns about security and terrorism, there has been an increasing focus on techniques that discover links and relations in data. Several efforts employ machine learning approaches to link analysis, but few consider mining meta-level patterns in higher-order links. In this work we focus on the discovery of such patterns in higher-order paths generated from supervised machine learning data. We use a dataset from the UCI machine learning repository for our analysis, and develop both theoretical and algorithmic approaches to enumerating and characterizing higher-order paths between attribute-value pairs. Based on statistical comparisons of distributions of higher-order path itemset frequencies, we discovered evidence that classes of instances in labeled training data may be separable based on the characteristics of higher-order paths.

These are preliminary results and we are researching more effective ways, both theoretical and algorithmic in nature that will aid us in mining higher-order path data. We are for example investigating more efficient algorithms to cope with the immense amount of data in bipartite graph representations of higher-order paths. As noted we are also exploring the formulation of closed-form representations of path data within the framework of systems of distinct representatives (SDRs).

In the experiments reported herein,

higher-order path patterns revealed differences among the instances of different classes. This technique may have applications in text mining as well. For instance, by considering a document or paragraph as an instance, we may determine higher-order path characteristics that aid in classifying text. In fact this approach may have an important application in security, counterterrorism and law enforcement. In order to evaluate this approach, in future work we plan to explore the application of these techniques on textual datasets such as the Enron email dataset.

Acknowledgements

This work was supported in part by NSF grant number 0534276 and National Institute of Justice, US Department of Justice grant numbers 2005-93045-PA-IJ and 2005-93046-PA-IJ. Points of view in this document are those of the authors and do not necessarily represent the official position or policies of the US Department of Justice or the National Science Foundation. Co-author William M. Pottenger wishes to thank His Lord and Savior Yeshua the Messiah for His grace. Thank you, Lord! Amen.

References

- [1] Bipartite Matching. <http://planetmath.org/encyclopedia/BipartiteMatching.html>
- [2] C. Borgelt and R. Kruse. Induction of Association Rules: Apriori Implementation. Proceedings of 14th Conference On Computational Statistics (COMPSTAT). Berlin, Germany, 2002.
- [3] S. Chakrabarti, B.E. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Mining the Link Structure of the World Wide Web, IEEE Computer, 1999.
- [4] H. Chen, and K.J. Lynch. Automatic construction of networks of concepts characterizing document databases. In IEEE Transactions on Systems, Man and Cybernetics 22 (5), 1992, pp. 885–902.
- [5] N. A. Chinchor. Overview of MUC-7/MET-2. Proceedings of the Seventh Message Understanding Conference (MUC-7), 1998.
- [6] R. Diestel. Graph Theory. Springer Press, 2000, ISBN 0-387-95014-1
- [7] P. Edmonds. Choosing the word most typical in context using a lexical co-occurrence network. In Proceedings of the Thirty-fifth Annual Meeting of the Association for Computational Linguistics, 1997, pp. 507-509.
- [8] Z. Galil. Efficient Algorithms for Finding Maximum Matching in Graphs. Computing Surveys, Vol. 18, No. 1, March 1986
- [9] R.V. Hauck, H. Atabakhsh, P. Ongvasith, H. Gupta, H. Chen, Using Coplink to analyze criminal-justice data, IEEE Computer 35 (3), 2002, pp. 30–37.
- [10] L. E. Holzman, T. A. Fisher, L. M. Galitsky, A. Kontostathis, and W. M. Pottenger. A Software Infrastructure for Research in Textual Data Mining. The International Journal on Artificial Intelligence Tools, 14 (4), 2004, pp. 829-849.
- [11] A. Kontostathis, and W. M. Pottenger. A framework for understanding LSI performance. Information Processing & Management, 42(1), 2006, pp. 56-73.
- [12] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI Repository of machine learning databases. <http://www.ics.uci.edu/~mllearn/MLRepository.html>. University of California, Irvine, Department of Information and Computer Science, 1998.
- [13] H. Schütze. Automatic word sense discrimination. Computational Linguistics 24 (1), 1998, pp. 97-124.
- [14] D. R. Swanson. Migraine and magnesium: eleven neglected connections. Perspectives in Biology and Medicine, 31(4), 1988, pp. 526-557.
- [15] T. Uno. Algorithms for Enumerating All Perfect, Maximum and Maximal Matchings in Bipartite Graphs. Lecture Notes in Computer Science, Vol. 1350. Proceedings of the 8th International Symposium on Algorithms and Computation, 1997, pp. 92 – 101, ISBN: 3-540-63890-3, Springer-Verlag, London, UK
- [16] J. H. Van Lint, and R. M. Wilson. A Course in Combinatorics. Cambridge University Press, 1993, ISBN: 0-521-42260-4
- [17] I. H. Witten, and E. Frank. Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
- [18] J. J. Xu and H. Chen. Fighting organized crimes: using shortest-path algorithms to identify associations in criminal networks. In Decision Support Systems 38(3), 2004, pp. 473-487.
- [19] J. Xu, W. B. Croft. Corpus-based stemming using co-occurrence of word variants. ACM Transactions on Information Systems 16 (1), 1998, pp. 61-81.
- [20] X. Zhang, M. Berry, and P. Raghavan. Level search schemes for information filtering and retrieval. Information Processing and Management 37 (2), 2000, pp. 313-334.
- [21] R. J. Mooney, P. Melville, L.R. Tang, J. Shavlik, I.C. Dutra, D. Page and V.S. Costa. Relational Data Mining with Inductive Logic Programming for Link Discovery. Proceedings of the National Science Foundation Workshop on Next Generation Data Mining, Nov. 2002, Baltimore, MD.
- [22] J. Sun, H. Qu, D. Chakrabarti, C. Faloutsos. Relevance Search and Anomaly Detection in Bipartite Graphs. SIGKDD Explorations, 7 (2), 2006, 48-55