

Link Prediction via Higher-Order Motif Features

Ghadeer Abuoda
College of Science and
Engineering, HBKU
Doha, Qatar
gabuoda@hbku.edu.qa

Gianmarco
De Francisci Morales
ISI Foundation
Turin, Italy
gdfrm@acm.org

Ashraf Aboulnaga
Qatar Computing Research
Institute, HBKU
Doha, Qatar
aaboulnaga@hbku.edu.qa

ABSTRACT

Link prediction requires predicting which new links are likely to appear in a graph. Being able to predict unseen links with good accuracy has important applications in several domains such as social media, security, transportation, and recommendation systems. A common approach is to use features based on the common neighbors of an unconnected pair of nodes to predict whether the pair will form a link in the future. In this paper, we present an approach for link prediction that relies on higher-order analysis of the graph topology, well beyond common neighbors.

We treat the link prediction problem as a supervised classification problem, and we propose a set of features that depend on the patterns or *motifs* that a pair of nodes occurs in. By using motifs of sizes 3, 4, and 5, our approach captures a high level of detail about the graph topology within the neighborhood of the pair of nodes, which leads to a higher classification accuracy. In addition to proposing the use of motif-based features, we also propose two optimizations related to constructing the classification dataset from the graph.

First, to ensure that positive and negative examples are treated equally when extracting features, we propose adding the negative examples to the graph as an alternative to the common approach of removing the positive ones. Second, we show that it is important to control for the shortest-path distance when sampling pairs of nodes to form negative examples, since the difficulty of prediction varies with the shortest-path distance. We experimentally demonstrate that using off-the-shelf classifiers with a well constructed classification dataset results in up to 10 percentage points increase in accuracy over prior topology-based and feature learning methods.

1. INTRODUCTION

Given a graph $G(V, E)$ at time t_1 , the *link prediction* problem requires finding which edges $\{e \notin E\}$ will appear in the graph at time $t_2 > t_1$ [27]. Predicting which new connections

are likely to be formed is a fundamental primitive in graph mining, with applications in several domains. In social media, friend and content recommendations are often modeled as link prediction problems [3]. Link prediction has also been used to detect credit card fraud in the cybersecurity domain [27, 49], to predict protein-protein interactions in bioinformatics [4], for shopping and movie recommendation in e-commerce [9], and even to identify criminals and hidden groups of terrorists based on their activities [6].

Traditionally, link prediction models rely on topological features of the graph, and on domain-specific attributes of the nodes (usually to induce a similarity function) [5]. Most topological features are based on common neighbors, i.e., they rely on the idea of ‘closing triangles’ [33]. More advanced approaches such as non-negative matrix factorization (NMF) and graph embeddings have also been tried recently [30, 46]. However, traditional topological features that rely on common neighbors, such as the Jaccard index and Adamic/Adar measure [1], have proven to be very strong baselines which are hard to beat [46].

These traditional features are not only effective, but also efficient to compute as they originate from triadic graph substructures. Fortunately, recent developments in algorithms and systems have improved our ability to efficiently count motifs with more than three nodes [2, 45]. Therefore, given the outstanding results of traditional topological features, it is natural to wonder about the predictive power of more complex features based on these higher-order motifs. In this paper, we show that higher-order motif features significantly improve the accuracy of link prediction models.

The present work focuses only on topological features, as node attribute features are domain- and application-specific, and are orthogonal in scope. As is common practice, we cast the link prediction problem as a binary classification task. We train a machine learning model on a sample of node pairs from the graph, where pairs with an edge between them represent a positive example, and pairs without an edge represent a negative one [12].

While developing our methodology, we faced several technical issues in extracting features for the link prediction problem that have not been explicitly addressed in the literature so far. Two issues deserve particular attention: how to generate motif features in a way that is consistent between training and testing, and how to select the negative examples for the dataset. For the first issue, the common practice is to remove a set of existing edges from the graph (the positive test set), and then train the classifier on the remaining edges. Here we propose an alternative based on adding a

set of negative examples (non-existing edges) to the graph when extracting the features. This version of the features consistently outperforms the former in terms of accuracy. For the second issue, we study the effect of node distance for negative examples on classification accuracy. We show that distance is an important factor that should be controlled for when creating a dataset (an under-appreciated fact in the link prediction literature [50]). This effect is consistent with the phenomenon of locality of link formation in growing networks [24].

The main contributions of this study are as follows:

- We show that more complex topological features based on higher-order motifs are powerful indicators for the link prediction problem in a variety of domains;
- These features improve the accuracy of standard classifiers by up to 10 percentage points over the state-of-the-art;
- We investigate how to correctly extract motif features and apply them to the link prediction problem;
- We re-examine the common practice of removing existing edges from the graph to create the classification dataset, and propose an alternative based on adding negative examples which provides better accuracy;
- We detail the effect of the distance of the pair of nodes for negative examples on the classification accuracy.

2. PROBLEM DEFINITION AND PRELIMINARIES

Consider graph $G(V, E_{t_1})$ at a given time t_1 , where V is the set of nodes in the graph and E_{t_1} is the set of edges that connect the nodes of the graph at that time. Link prediction aims to predict which new edges are likely to appear at time $t_2 > t_1$, i.e., to predict the set $\{e : e \notin E_{t_1} \wedge e \in E_{t_2}\}$. Clearly, the assumption is that the set of nodes V does not change in time. In addition, we assume the graph G is undirected and unweighted.

While the real application of link prediction involves time, very often testing prediction algorithms in these conditions is not straightforward, mostly due to the unavailability of the history of the evolution of the graph structure. Therefore, in most cases, link prediction is cast as a standard binary supervised classification task [6]. In this scenario, each data point corresponds to a pair of nodes (u, v) in the graph, and the label

$$L(u, v) = \begin{cases} +1 & (u, v) \in E, \\ -1 & (u, v) \notin E. \end{cases}$$

To build a dataset for classification, we need a training set E_{TR} and a test set E_{TS} . Usually, a small fraction of the edges in the graph $E_{TS}^+ \subset E$ are kept aside for testing, and the prediction model is trained on (part of) the rest of the edges in the graph $E_{TR}^+ \subseteq E \setminus E_{TS}^+$. Often, to obtain a better statistical evaluation of the predictor, this procedure is repeated via cross-validation.

While the edges in the graph can be used for the positive examples, a binary classifier also needs negative examples. We can extract a set of negative examples by sampling pairs of nodes from the graph which are not connected by an edge, i.e., $E_{TR}^- \subset \{(u, v) : u, v \in V \wedge (u, v) \notin E\}$, and similarly for the test set E_{TS}^- . We call these pairs *negative edges*.

The major challenge of this approach is to choose suitable features for the classification task. We address this point in the next section, where we describe the motif features we propose. Our hypothesis is that more complex topological features, which look deeper in the structure of the neighborhood of two nodes, are more powerful predictors of link formation than features that rely on shallow structural information such as common neighbors. However, an equally important problem is how to create the classification dataset, and in particular, how to sample the negative edges. This problem has not received much attention in the literature so far, but we show that it has a significant impact on classification accuracy.

2.1 Motifs

Motifs are small, connected, non-isomorphic subgraphs which appear in a larger graph [31, 42]. Each k -motif represents a topological pattern of interconnection between k nodes in a graph. Figure 1 shows all the 29 possible motifs for $k \in \{3, 4, 5\}$. We denote each motif as ‘mk.n’ where k is the number of nodes in the motif and n is an ordinal number which identifies the specific edge pattern in the motif (we use this naming of motifs throughout the paper).

Motifs differ from the related concept of graphlets since they are not induced (i.e., they can be partial subgraphs), while graphlets are induced subgraphs (i.e., there is a bijection between the graphlet and its occurrence in a graph) [37]. Motifs are often associated with a specific frequency threshold, which needs to be higher than a null-model randomized graph. We do not impose such restriction in the current work, and let the machine learning algorithm determine whether a particular motif is significant for our task. The concept of motifs was originally introduced by [31], which showed their modeling power in biological networks.

Motifs have been shown to be a powerful graph analysis tool in previous work. The motif profile, the frequency distribution of the motifs in a graph, is used as a ‘fingerprint’ of a graph [32]. Therefore, the usefulness of motifs to capture the macro structure of a graph is well established [48]. However, for our purpose, we are more interested in their ability to capture the micro and meso structure (i.e., the neighborhood) of the graph. Given that features based on triangle closing (i.e., common neighbors) are already quite

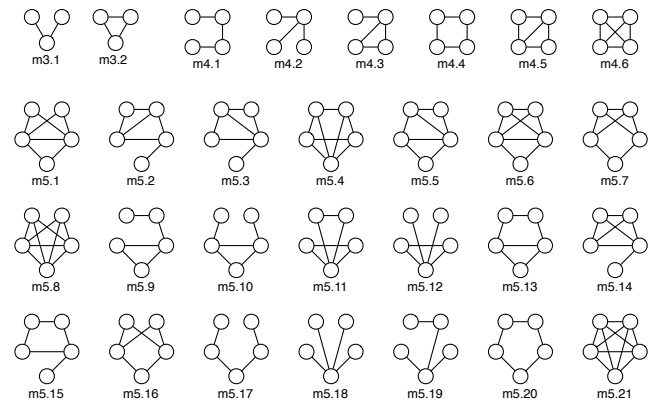


Figure 1: All the possible undirected k -motifs for $k \in \{3, 4, 5\}$. There are 2 different 3-motifs, 6 different 4-motifs, and 21 different 5-motifs.

effective, we expect features based on k -motifs for $k > 3$ to hold even more predictive power.

Counting k -motifs is an expensive operation, as their number grows exponentially in k . However, thanks to recent advances in both algorithms and systems, we are now able to count k -motifs on graphs with millions of edges for values of k of 5 or more [8, 45]. We leverage this capability to capture complex topological features for the link prediction task, and go beyond the simple triangle-based features that have been traditionally used.

3. MOTIF FEATURES

As mentioned in the previous section, we approach the link prediction problem as a binary classification problem, where the features used for classification are based on the motifs of the graph. In this section, we provide details on feature extraction and on sampling the example edges to create the dataset that is used for training and testing.

The features in our model correspond to the number of occurrences of an edge (positive or negative) in the classification dataset within different k -motifs. That is, for each example edge in the classification dataset, we enumerate the k -motifs that the edge is part of, and then count the occurrences of each different motif. In this paper, we use 3-, 4-, and 5-motifs. Motifs of even higher order are prohibitively expensive to compute for large graphs, and we experimentally demonstrate high prediction accuracy with $k \in \{3, 4, 5\}$. The number of k -motifs is the number of possible shapes for a connected graph with k nodes. The possible motifs for $k = 3, 4$, and 5 are shown in Figure 1. There are 2, 6, and 21 motifs for $k = 3, 4$, and 5 , respectively, and this is the number of features we generate for each k .

When generating the motif features, the example edge plays an important role. Depending on its class, the edge might exist in the original graph ($(u, v) \in E \implies L(u, v) = +1$), or it might be a sampled negative pair, so that the edge does not exist in the original graph ($(u, v) \notin E \implies L(u, v) = -1$).

3.1 Equal Treatment of Positive and Negative Examples

It is of paramount importance to treat both positive and negative example edges in the same way with respect to feature extraction, especially when dealing with the test set. To exemplify why this is important, imagine using $k = 3$ and not addressing this issue. The two possible features are then the wedge (or open triangle) and the closed triangle (m3.1 and m3.2 in Figure 1). Positive example edges will have a mix of both features, but negative example edges will never appear in a closed triangle, by construction. Therefore, this way of extracting features leaks information about the class into the features themselves. This leakage is clearly an issue for the test set, but in order for the features to be meaningful, we need to apply the same extraction process to both the training set and the test set.

To solve this issue we have two possible options: (i) remove positive edges from the motif, which we denote **RMV**, or (ii) insert negative edges into the motif, which we denote **INS**. The former option corresponds to the traditional way of handling link prediction as a classification task, where a set of (positive) edges are withheld from the model. The latter is a novel way of handling the feature extraction that has not been considered previously. It corresponds to asking

the following question: “If this edge was added to the graph, would its neighborhood look like other ones in the graph?”

In the first method, **RMV**, we remove the example positive edge from the graph and extract the features by looking at motifs that contain both endpoints of the removed edge. We match the two endpoints of the example edge rather than the edge itself because otherwise **RMV** would never be able to find any relevant motif in the graph, as the example edge is no longer present in the graph. The features for negative edges are computed in a similar manner, by looking at the motifs containing both endpoints of the negative pair. In this case, no modification to the graph is needed for negative edges.

By following this methodology, a number of motifs will never appear as features (e.g., fully connected cliques). In addition, an example edge never contributes to producing the motifs that it is part of. An example for $k = 4$ is shown in Figure 2. Assume green edges are positive examples, red edges are negative examples, and black edges are part of the graph but are not part of the classification dataset (i.e., they have not been sampled). Additionally, dashed edges are removed from the graph. In this case, positive edge (1, 2) is removed from the graph and negative edge (2, 3) is not inserted. Removing edge (1, 2) changes the motifs in this neighborhood. For example, motif m4.5 does not appear in the neighborhood even though edge (1, 2) was part of an instance of this motif.

Let us go briefly through the example to show how motif extraction and counting works. Given that we remove example edges from the graph, motifs m4.4, m4.5, and m4.6 do not appear in the graph as no “square” shaped closed path exists. Motif m4.3 can only appear once as it has the same number of edges as the graph. Motif m4.2 can only appear once, as there is only one node with degree 3 in the graph (node 0) and the motif is symmetric. Motif m4.1 appears twice as motifs are edge-induced rather than node-induced, so there are two possible matches: “2-0-1-3” and “0-1-3-0”. The resulting counts for both the positive and negative edge are the same.

In the second method, **INS**, we insert negative example edges into the graph before extracting and counting motifs. No modification to the graph is needed for positive example edges. After inserting the negative example edges, we count the motifs for positive and negative edges in the same way.

In the **INS** methodology, all motifs can appear as features, and an example edge contributes to all the motifs it is part of. For example, Figure 3 shows the same graph as Figure 2, but now the negative example edge (2, 3) is added to the graph. All six motifs appear as features for the negative edge, and each feature of an example edge (positive or negative) corresponds to a motif which includes the edge itself. Note that this is different from **RMV**, as now we can restrict counting only to motifs which the *edge* is part of, rather than just looking at the endpoints. We experimentally compare **RMV** and **INS** in Section 4.

When using the **INS** method, we insert all of the negative edges in the graph before doing any feature extraction. Sampling a negative edge in the neighborhood of a positive one changes the extracted features, as shown in Figure 3. That is, the extracted motifs are not fully independent of the sampling. If a negative sample edge happens to be part of the same motif as a positive sample edge, each edge will affect the motif counts of the other. While this is not desirable, we verify that the occurrence of these cases in practice

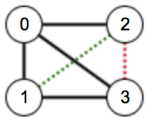
			Features					
Graph	Edge	Class	m4.1	m4.2	m4.3	m4.4	m4.5	m4.6
	1-2	Positive	2	1	1	0	0	0
	2-3	Negative	2	1	1	0	0	0

Figure 2: Motif features when positive example edges are removed from the graph (RMV).

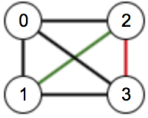
			Features					
Graph	Edge	Class	m4.1	m4.2	m4.3	m4.4	m4.5	m4.6
	1-2	Positive	0	2	1	0	1	1
	2-3	Negative	3	2	2	1	2	1

Figure 3: Motif features when negative example edges are inserted into the graph (INS).

is very rare, so they do not affect the learning process in any significant way.

Let us walk through an example count for Figure 3. Take motif m4.3, for instance. There are 2 possible distinct matchings of it in the graph. Only one of the two matches includes the positive edge (1, 2) and both matches include the negative edge (2, 3). A similar reasoning applies to the other motifs. Note that for INS the resulting motif counts are different for the positive and negative edges.

3.2 Sampling Negative Edges

Another important question, independent of choosing RMV or INS, is how to sample the edges for the classification dataset. For positive example edges, uniform random sampling is an adequate solution, reflecting the assumption that no edge is easier or harder to predict than another. For negative example edges, however, it is easy to imagine that an edge connecting two nodes in completely different regions of the graph is less likely to occur than one connecting two nodes in the same region. Therefore, the distance between the pair of sampled nodes can play an important role. For this reason, we choose to control this parameter.

We sample negative edges based on the shortest-path distance between the endpoint nodes. We choose to use a mix of nodes with distance $d = 2$ and $d = 3$ hops. Figure 4 shows examples of sampling negative edges with these two distances. In most of the experiments, we use a 50/50 split between negative example edges at distance 2 and 3. However, we also analyze the effect of the distance on classification accuracy by changing the ratio between these two sub-classes.

When building the classification dataset, we sample an equal number of negative and positive edges. This decision allows us to use simple classification measures, such as accuracy, without the issues that arise due to class skew. In a typical graph, most pairs of nodes would not have an edge

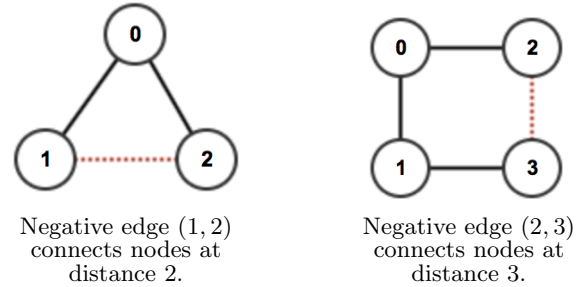


Figure 4: Two negative edges (dashed red lines) at different distances.

connecting them, so the negative class would be much larger than the positive class. However, as we are only interested in the relative performance of the features, and because we use off-the-shelf classifiers, we prefer to create a balanced classification dataset.

3.3 Classification

The discussion up to this point enables us to create a classification dataset with an equal number of positive and negative examples. The negative examples are pairs of nodes with a shortest-path distance of 2 or 3 (typically split 50/50 between the two distances). Each example has a set of features based on its k -motifs, where k is 3, 4, 5, or a combination thereof.

This classification dataset can be used as input to any machine learning classification algorithm. We experiment with several off-the-shelf classifiers, and we show in the following section that this method of creating a classification dataset combined with a powerful classification model leads to very high prediction accuracy.

Table 1: Basic statistics about the graph datasets.

Graph	V	E	Avg. degree	Diameter
Amazon	334 863	925 872	5.530	47
CondMat	22 015	58 595	3.025	36
AstroPh	18 771	198 050	21.102	14

Table 2: Number of positive and negative edges sampled for the classification dataset for each graph.

Graph	# Positive Edges	# Negative Edges
Amazon	20 000	20 000
CondMat	2000	2000
AstroPh	5000	5000

4. EXPERIMENTS & EVALUATION

This section describes how we apply and evaluate the motif features for link prediction on three different graphs. In particular, we look at the predictive power of motif features for different motif sizes. We also compare the different methodologies for feature extraction described in Section 3.

4.1 Datasets

We use three real-world graphs coming from different domains: Amazon, CondMat, and AstroPh. The three graphs are from the Koblenz Network Collection.¹ Table 1 shows basic statistics about these graph datasets.

The first graph represents the co-purchase network of products on Amazon. It is the graph upon which the “customers who bought this also bought that” feature is built. The nodes are products, and an edge between any two nodes shows that the two products have been frequently bought together. In this application domain, link prediction tries to model the creation of an association between two products. In other words, it can help identify hidden relationships between products, which can then be used in a customer-facing recommendation system.

The second dataset, CondMat, is a bipartite graph which represents a subset of authorship relations between authors and publications in the arXiv condensed matter physics section. Nodes are authors (first set) or papers (second set). An edge represents that an author has written a given paper. Link prediction can identify whether an author is a likely contributor to a research paper, for instance, to identify missing records.

The third and final dataset, AstroPh, is a collaboration graph. In particular, it contains data about scientific collaboration between authors in the arXiv astrophysics section. Each node in the graph represents an author of a paper, and an edge between two authors represents a common publication. Predicting future collaborations between authors based on their previous publications is a well-known application for link prediction.

¹<http://konect.uni-koblenz.de>

4.2 Experimental Settings

For each graph, we extract a classification dataset for which we compute node pair features. We extract a uniform sample of edges from each graph as positive examples. For negative examples, we extract pairs of nodes from the graph which are at distance 2 or 3 hops. Table 2 shows the number of examples chosen from each graph. As already mentioned, to assess the predictive power of the motif features in a simple setting, we build balanced classification datasets.

We extract the motifs that the example edges in the classification dataset occur in by using the Arabesque parallel graph mining framework [17, 45]. We then group by motif, count the occurrences, and finally normalize the counts to create a feature vector for each example edge which represents the motif distribution of the neighborhood of the edge.

To train the classification models we use the scikit-learn Python library [36]. We train naïve Bayes (NB), logistic regression (LR), decision tree (DT), k-nearest neighbors (KNN), gradient boosted decision trees (GB), and random forest (RF) models. All the classification performance results are computed via 10-fold cross-validation.

Baselines. We use several baselines, which we divide into two types. The first type is using traditional topological features such as triangle closure or paths. We compare our features against common neighbors [34], Jaccard coefficient [27], Adamic/Adar measure [1], Preferential Attachment [33], rooted PageRank [43], and Katz index [20].

The second type of baseline includes more complex techniques such as matrix decomposition, which is commonly used in link prediction, and deep learning. For matrix decomposition, we use the scores obtained from a non-negative matrix factorization (NMF) trained on the graph with positive edges removed. This methodology follows the common approach used in the literature [30]. We use the NMF algorithm available in scikit-learn, and use 100 factors for the decomposition. For deep learning, we compare against a recent state-of-the-art graph neural network framework for link prediction called SEAL [51]. SEAL uses subgraph extraction around the example edge to extract latent features, learned via a neural network. This framework has been shown experimentally to outperform other existing deep learning methods. Given that SEAL uses subgraph-based latent features, which are similar in spirit to the motifs used by our methodology, comparing against SEAL is of particular interest and importance.

4.3 Evaluation Metrics

We evaluate the classification task via the following three metrics:

- Accuracy (ACC): the fraction of examples correctly classified (true positives and true negatives) over the total number of examples (N).

$$ACC = \frac{TP + TN}{N}$$

Given that the classification datasets are balanced, accuracy is a reasonable measure of performance. Better classifiers obtain higher accuracy.

- Area Under the Curve (AUC): the area under the Receiver Operating Characteristic (ROC) curve generated from the scores produced by the classifiers. This measure

Table 3: Classification performance on Amazon (RMV).

Metric	Classifier	Features			
		$k = 3$	$k = 4$	$k = 5$	Combined
ACC (%)	NB	57.6	52.4	52.7	52.0
	LR	56.5	59.4	68.0	64.4
	DT	57.6	69.4	70.8	70.6
	KNN	51.5	69.9	71.4	71.0
	GB	58.0	73.3	76.6	76.9
	RF	57.6	71.6	76.3	77.0
AUC	GB	0.58	0.72	0.76	0.76
	RF	0.58	0.72	0.76	0.77
FPR	GB	0.11	0.30	0.26	0.27
	RF	0.11	0.32	0.27	0.27

represents the probability that a classifier will rank a randomly chosen positive example higher than a randomly chosen negative one. Better classifiers obtain higher AUC.

- False Positive Rate (FPR): the ratio between the number of negative edges wrongly classified (false positives) and the total number of negative edges in the dataset.

$$FPR = \frac{FP}{FP + TN}$$

This measure is useful to understand the effect of the graph distance of the negative examples on the classification task. Better classifiers obtain lower FPR.

Table 4: Classification performance on Amazon (INS).

Metric	Classifier	Features			
		$k = 3$	$k = 4$	$k = 5$	Combined
ACC (%)	NB	57.7	57.2	52.7	53.6
	LR	62.5	67.7	70.0	67.5
	DT	67.9	66.9	69.6	71.0
	KNN	63.6	66.0	64.0	65.0
	GB	68.2	75.0	76.6	79.4
	RF	68.0	74.8	77.0	79.6
AUC	GB	0.69	0.74	0.76	0.80
	RF	0.68	0.75	0.78	0.80
FPR	GB	0.25	0.25	0.25	0.18
	RF	0.23	0.23	0.21	0.18

4.4 Removing Positive Edges vs. Inserting Negative Edges

Tables 3 and 4 show the classification results of the two feature extraction methods (RMV and INS, respectively) on the Amazon dataset (the largest one). We report the results for all classifiers when using features based only on motifs of size $k = 3$, only on motifs of size $k = 4$, and only on motifs of size $k = 5$. We also report in the last column the results when using all three sets of features together in one feature vector (total of 29 features).

By looking at the difference between the two tables (first six rows), it is clear that INS consistently has higher accuracy than RMV. The difference grows smaller as we add more complex features by increasing k . However, for the two best classifiers (GB and RF), INS still results in approximately 3 percentage points higher accuracy than RMV even when using the combined features. The simpler classifiers do not seem able to exploit the full predictive power of the motif features.

Table 5: Classification performance of a Random Forest (RF) classifier with combined motif features when using different feature extraction methods (RMV vs. INS).

Method	Graph	ACC	AUC	FPR
RMV	Amazon	0.770	0.77	0.27
	CondMat	0.790	0.79	0.04
	AstroPh	0.840	0.84	0.30
INS	Amazon	0.796	0.80	0.18
	CondMat	0.960	0.96	0.04
	AstroPh	0.965	0.97	0.02

On the other hand, GB and RF are advanced classifiers that can exploit this predictive power.

Tables 3 and 4 report AUC and FPR for the two best classifiers. The AUC and FPR are very similar, and the two classifiers are almost indistinguishable. As expected, the more complex motif features (i.e., larger k) work better, and the combination of all three sets of features is usually the best. For ease of presentation, henceforth we report results only using the RF classifier, but results using GB are similar.

Figure 5 reports the accuracy numbers for RF on all three datasets, for both feature extraction methods. The results are consistent with what we observed above: INS is consistently better than RMV. Interestingly, INS with just 3-motif features performs better than RMV with the combined motif features on the CondMat and AstroPh datasets.

We perform a statistical test to compare the classification accuracy of the two methods, INS and RMV. We obtain 100 different samples of the accuracy for each method by training the RF classifier using different seeds for the pseudo-random number generator. We use Student’s t-test to compare the results, and we are able to reject the null hypothesis that the two methods have the same average performance at the $p = 0.05$ significance level. We conclude that the accuracy of the RF classifier with INS feature extraction is better than the one with RMV, and the difference is statistically significant. It is clear that the INS feature extraction method is superior, and we return to the reason behind this in Section 4.6.

More complex motif features perform better, with the combination of all motif sizes outperforming each individual size. The latter result might seem surprising, as one would expect the 5-motif features to supersede the smaller ones. Nevertheless, consider that our 5-motif features do not encode positional information, i.e., we do not know in which part of the 5-motif the edge appears. Smaller features can provide this information, thereby supplementing the 5-motif features. For example, if a sample edge presents feature m5.3 from Figure 1, it could be any of that motif’s edges. But if the edge also presents no instances of feature m3.2, then we can pinpoint it to be the only the edge in the m5.3 motif that is not involved in any triangle, i.e., the bottom right edge.

Finally, Table 5 reports ACC, AUC, and FPR for RF on all datasets for the two different feature extraction methods when using the combined motif features. The mix of RF, combined motif features, and INS feature extraction is the one that performs consistently on top. Therefore, we use it when comparing our proposal with baseline methods in the following section.

Table 6: Accuracy of the RF classifier (%) when using combined motif features (with **INS**) vs. baseline classifiers.

Features	Amazon	CondMat	AstroPh
Common Neighbors	64.6	78.6	81.2
Jaccard Coefficient	61.7	81.1	85.2
Adamic/Adar	61.5	74.7	75.0
Preferential Attachment	55.0	61.2	64.2
Rooted PageRank	53.2	62.0	65.0
Katz Index	60.0	55.0	59.0
Topological Combined	73.0	86.9	87.0
NMF	52.0	54.0	53.5
NMF + Topological Combined	73.0	85.9	89.0
SEAL	69.0	81.3	80.3
SEAL + node2vec Embeddings	62.8	77.2	82.0
Motif Combined (INS)	79.6	96.0	96.5

4.5 Comparison with Baselines

To compare against the baseline topological features proposed in prior work, we train a Random Forest classifier on each of these features. We also train a Random Forest classifier on a combination of all of these features. The top part of Table 6 reports the accuracy of these classifiers. For comparison, the accuracy of the RF classifier trained on the combined motif features extracted via **INS** is reported in the last row of the table. The first four rows of the table show simple neighborhood-based topological features, namely common neighbors, Jaccard coefficient, Adamic/Adar measure, and Preferential Attachment. The next two rows show path-dependent topological features, namely rooted PageRank [43] and Katz index [20]. For the PageRank, we use the standard value for the damping parameter $\alpha = 0.85$. For the Katz index, we optimize the value of the β parameter and we report the highest accuracy obtained, which in our case is for $\beta = 0.1$. The accuracy of the topological features is in the range 55–85%, with the neighborhood-based and path-dependent features exhibiting similar accuracy. One important takeaway from the table is that combining all topological features into one feature vector results in the best accuracy in all cases. This is expected since each of these features captures different information about the graph, and a powerful classifier such as RF is able to exploit all of this information. Thus, the Topological Combined row in Table 6 can be viewed as the best possible accuracy with current state-of-the-art topological features: 73% on Amazon and around 87% on the other two datasets. However, the motif features achieve much higher accuracy (last row). Specifically, they are 7 to 10 percentage points better in accuracy, which is quite significant, especially given that advanced features extracted via graph embeddings and deep learning reportedly struggle to beat the traditional topological features [46].

Next, we turn our attention to feature learning methods that allow the model to determine by itself which features are important for link prediction. As mentioned earlier, we focus on two popular approaches: non-negative matrix factorization (NMF) and deep learning. Interestingly, the NMF approach [30] is not very competitive. Table 6 shows that the accuracy of NMF is much lower than RF with motif features. We hypothesize that the method requires more parameter optimization (e.g., tuning the number of factors used and the regularization parameters). In any case, the

gap between NMF and straightforward topological features is quite large when used out of the box, which is quite disappointing. Moreover, adding the NMF features to the topological ones and training a classifier on this combined set of features does not improve accuracy by much (only the AstroPh dataset sees some improvement).

Finally, we compare our model with SEAL [51], a recent link prediction framework which uses deep learning (graph neural networks). SEAL learns latent subgraph features that are then used for link prediction. We test the framework on the three datasets by training and testing it on the same sampled edges used for the other classifiers. Interestingly, SEAL only achieves around 70% accuracy on Amazon and 80% accuracy on the other two datasets. The latent subgraph features learned by SEAL are expected to capture a signal similar to the topological features. SEAL learns on one- or two-hop subgraphs extracted around the tested edge, which is somewhat equivalent to looking into common neighbors or Adamic/Adar, regardless of the prediction model (neural vs. traditional). However, the accuracy achieved by SEAL is lower than with the topological features combined.

We also test combining the subgraph features with node representations learned via node2vec [16], as suggested by the authors of SEAL. The accuracy with the node2vec embeddings did not improve on average, and actually dropped for two of the datasets. One interpretation of these results is that the node2vec embeddings might actually introduce noise in the node representations, by looking too far into the neighborhoods of the example edges (e.g., if the length of the random walks is not appropriately tuned).

Thus, the overall takeaway from Table 6 is that RF with motif features is more accurate than all baselines, both traditional topological-based ones and more recent NMF and deep learning ones.

Feature importance. We analyze the motif features that are most predictive for the classification task. Figure 6 shows the relative importance of the features as inferred by the Random Forest classifier. Feature importance for the RF classifier is defined by Mean Decrease in Impurity (MDI) [38]. The MDI importance of a feature is calculated by measuring how effective the feature is at reducing uncertainty when creating decision trees within the RF classifier.

In most cases the distribution of feature importance is quite skewed, with a few features constituting the backbone of the predictive model. Interestingly, the most predictive

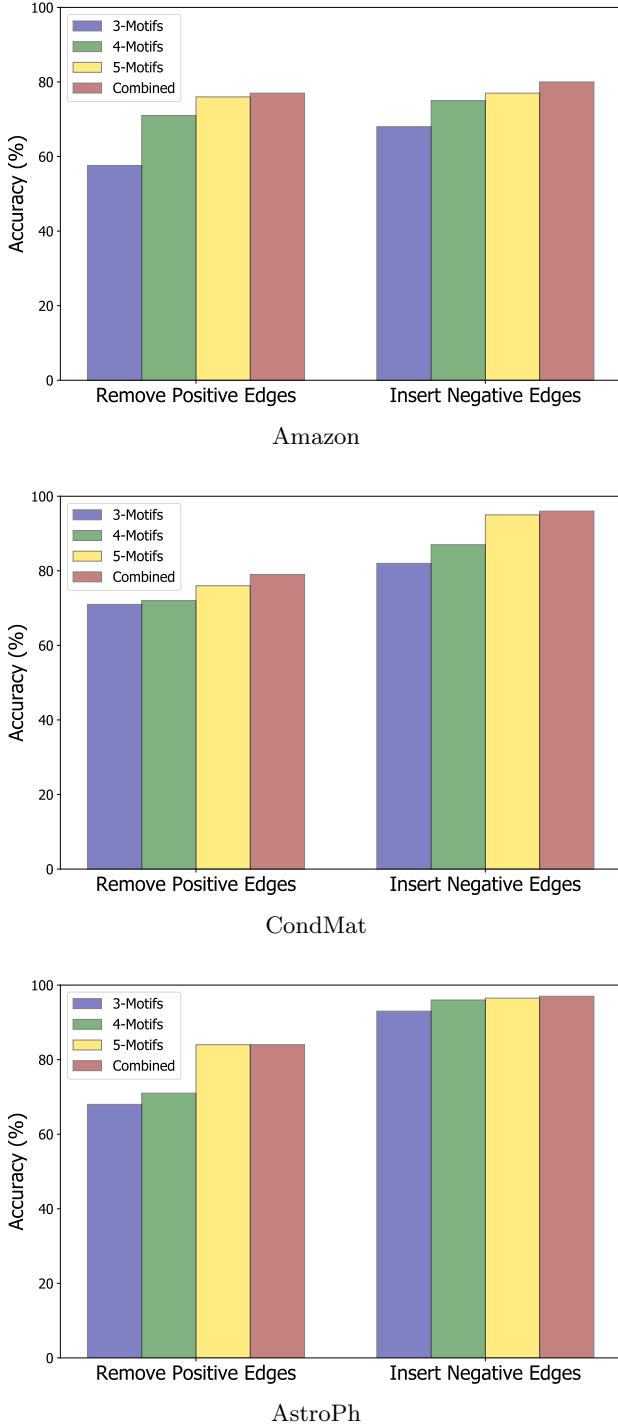


Figure 5: Classification accuracy of a Random Forest (RF) classifier when using different motif features and different feature extraction methods (RMV vs. INS).

Table 7: Earth Mover’s Distance (EMD) and Kullback–Leibler Divergence (KLD) between the distribution of motifs in the original graph and the one obtained by each feature extraction method, RMV and INS. A smaller distance indicates that the given feature extraction method is more faithful to the original graph.

Graph	EMD		KLD	
	RMV	INS	RMV	INS
Amazon	0.119	0.011	0.007	0.001
CondMat	1.106	0.161	0.533	0.012
AstroPh	0.050	0.529	0.001	0.066

feature is always a 5-motif one, which is another indication of the predictive power of deeper structural features. However, the most important feature changes from dataset to dataset, and might be domain specific.

As an illustration of the utility of feature importance, we train the RF classifier with just the top-10 features by importance for each dataset, and we observe that the accuracy obtained is almost identical to the full model. This result indicates that feature selection could be used to identify a few important motifs in a dataset, and extract only those, which would reduce the computational burden of counting all the motifs.

Overall, these results prove the predictive power of higher-order motif-based features for link formation. The rest of the experimental section is devoted to three more questions related to motif feature extraction and negative edge sampling. First, we shed some insight about why INS performs better than RMV. Second, we show the importance of choosing the right negative examples, an important factor which has been mostly overlooked in the literature thus far. Finally, while not the main focus of the current paper, we show that motif extraction is quite scalable and thus easily applicable to medium sized graphs.

4.6 Motif Distribution: RMV vs. INS

Let us now look at the reason why INS outperforms RMV for feature extraction. Consider that both feature extraction methods change the original motifs of the graph, as they alter the graph structure during feature extraction. One hypothesis is that the method which alters the structure the least is better, as the motif patterns it learns are also the closest to the ones found in the original graph. To test this hypothesis, we compute the motif distribution in the original graph and in the modified graphs resulting from the modifications done by RMV and INS (i.e., with a fraction of edges removed or added). We compute the motif distribution for $k = 3$ and 4 for the whole graph, and compute the distance between the original distribution and the distribution obtained by RMV and INS. We use two different distance functions to perform the comparison: Earth Mover’s Distance (EMD) [23] and Kullback–Leibler Divergence (KLD) [22]. Table 7 reports the results. If our hypothesis is correct, then INS should have a smaller distance than RMV. This is indeed the case for two out of three graphs, for both distance functions, which gives us confidence that our hypothesis is a step in the right direction. However, AstroPh behaves differently, with RMV having a larger distance than INS. Therefore, we cannot draw a definitive conclusion, and further study is necessary to fully understand the difference between these

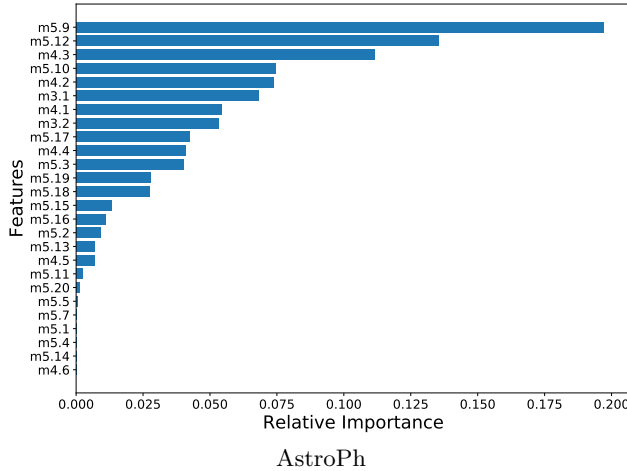
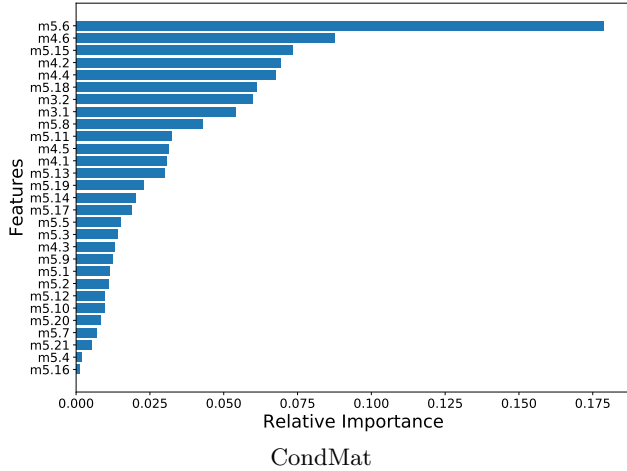
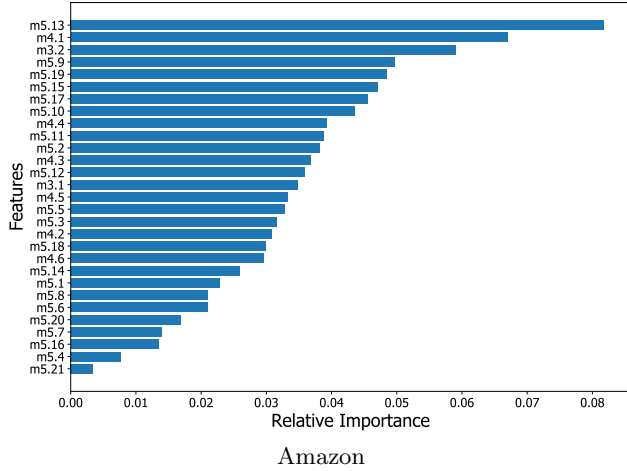


Figure 6: Feature importance across the three datasets, as inferred from the Random Forest model. In all three datasets the most important feature is a 5-motif, however the specific motif varies by dataset.

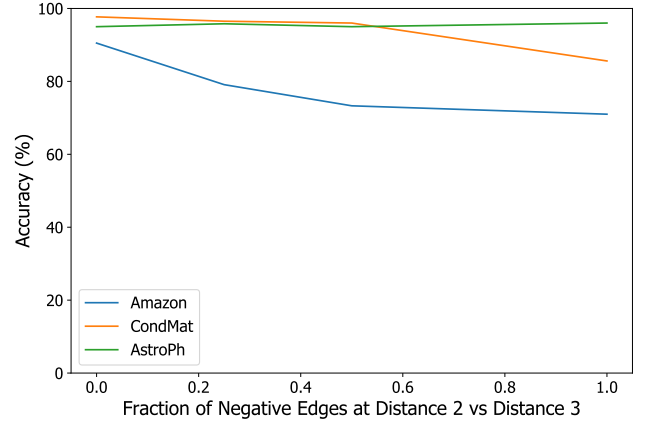


Figure 7: Classification accuracy as a function of the fraction of negative examples at distance 2 (vs. distance 3).

two motif feature extraction methods.

4.7 Effect of Distance on Negative Edges

In this experiment we explore the effect of the distance between the node pairs that constitute the negative examples on the accuracy of the classifier. For each graph, we create different classification datasets by varying the composition of the negative class: from containing only negative edges at distance 3 to containing only negative edges at distance 2. We use the fraction of negative edges of the sub-class at distance 2 as the independent variable in the plots (the rest of the edges are at distance 3). We keep the total number of examples fixed to maintain the balance between positive and negative classes.

Figure 7 shows the classification accuracy for each setting. For both Amazon and ContMat, the edges at distance 2 are harder to classify correctly, which produces a significant decline in the accuracy as we increase the fraction of edges at distance 2. Conversely, the accuracy on AstroPh does not seem affected. The same pattern can be seen in Figure 8, which reports the false positive rate. The figure explains the cause of the decrease in accuracy: as we decrease the average distance of the negative examples, the classifier produces more false positives. The higher the fraction of negative examples at distance 2, the higher the rate of misclassification for the negative class. Again, AstroPh is the exception in our experiments.

We conclude that classification accuracy can be affected by the choice of the negative examples, and that the distance of negative samples should be carefully controlled. In general, negative examples with smaller distances are harder to classify correctly, as they are more likely to be false positive. However, the effect remains dataset-specific.

4.8 Scalability of Motif Extraction

Lastly, we show the time taken to extract 5-motifs on our largest graph, Amazon. These times were measured while running Arabesque [45] on 4 compute nodes with 8 cores and 256GB of memory each.

We measure the extraction time while varying the number of edges. The results can be seen in Figure 9, for 2000, 20 000, and 40 000 edges. The times taken are around, 4, 6, and 11 hours, respectively. The lower efficiency for 2000

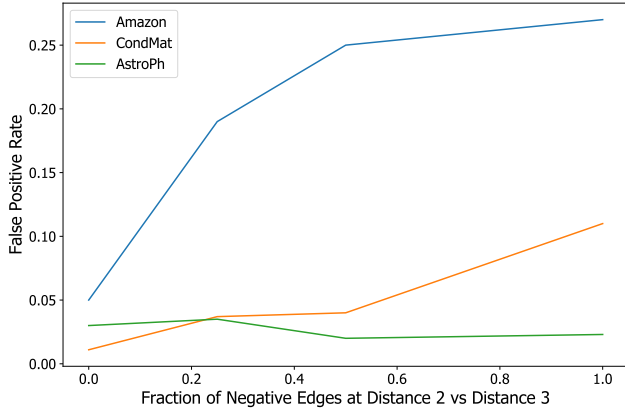


Figure 8: False positive rate as a function of the fraction of negative examples at distance 2 (vs. distance 3).

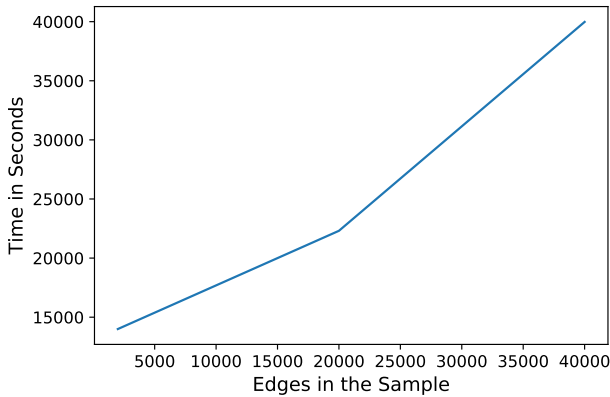


Figure 9: Time required to extract 5-motifs (m5.*) for different sample sizes on the Amazon graph.

edges (500 edges/hour) is due to startup costs of the motif-counting framework, which can be amortized over a larger number of edges (3600 edges/hour for 40 000 edges).

5. RELATED WORK

There are two main branches of research that are relevant to the current work: graph motifs and link prediction.

5.1 Graph Motifs

Motifs are patterns of connectivity that occur significantly more frequently in the given graph than expected by chance alone [48]. Graph motifs have numerous applications, for example, they have been used to classify graphs into “superfamilies” [41], and they have been used in combination with machine learning techniques to determine the most appropriate network model for a given real-world network [44]. Palla et al. [35] also show that 4-cliques (fully-connected motifs of 4 vertices) reveal community structure in word associations and protein-protein interaction graphs.

In several social media analysis studies [18, 19, 47], graph motif detection and enumeration are used to characterize graph properties statistically. The significance of motifs is typically assessed statistically by comparing the distribution of subgraphs in an observed graph with the one found

in a randomized graph. The randomization process is designed to alter the structure of the graph while preserving the number of nodes, edges, and the degree distribution [48]. One of the important reasons why graphs in the real world have more motif structure than the randomized version is that real-world graphs are constrained by particular types of growth rules, which in turn depend on the specific nature of the graph. In this paper, we aim at leveraging this property to learn which specific motifs are predictive of link presence.

5.2 Link Prediction

Prior work on link prediction can generally be classified into three broad categories: unsupervised methods, supervised methods, and feature learning methods. Link prediction methods can also be orthogonally classified by the type of information they rely on: node properties or structural properties. The two methodologies can always be combined when both types of information are available.

Unsupervised methods. In most unsupervised methods, a heuristic is chosen and used to rank node pairs in the graph, with a higher rank indicating a higher likelihood of a link existing between the node pair [13, 29]. The heuristic is typically a similarity measure, and can be based on application-specific node attributes or on the graph topology. Examples of node attributes used in unsupervised methods include user profiles in social networks [49] and attributes of publication records in academic co-authorship graphs, such as the topics a researcher works on [10] or the conferences they submit to [26]. While node attributes can achieve a high degree of accuracy for link prediction, they are domain- and application-specific, and cannot be easily generalized. In contrast, features based on graph topology are more general and directly applicable to any graph.

Topological features that are used in unsupervised link prediction are typically related to local (neighborhood) or global (path) properties of the graph. Neighborhood-based features capture the intuition that a link is likely to exist between a pair of nodes if they have many common neighbors. The simplest neighborhood-based feature is to count common neighbors (i.e., open triangles) [33]. More advanced features include some form of regularization of the count, such as the Jaccard coefficient of the two sets of neighbors, and the Adamic/Adar index [1], which discounts the contribution of high-degree neighbors, or preferential attachment [7], which gives a higher likelihood to links between high degree vertices. Local similarity indices are easy to compute, and scale well to large graphs.

Conversely, path-based features look at the global graph structure. A representative path-based feature is the classic Katz index [20], which counts the number of paths between two nodes, giving a higher weight to shorter paths. Other methods such as hitting time, commute time, and rooted PageRank use random walks on the graph to derive the similarity of two nodes. Global similarity indices typically provide better predictions than local indices, but are more expensive to compute, especially in large graphs. For a detailed survey of unsupervised link prediction methods, see the works by Getoor and Diehl [15] and Lichtenwalter et al. [28].

Several studies indicate that unsupervised methods are fundamentally unable to cope with dynamics, imbalance, and other complexities of real-world graphs [6, 28]. However,

similarity indices can easily be used by supervised methods as features for a machine learning model.

Supervised methods. In supervised methods, link prediction is usually cast as a binary classification problem. The target class label indicates the presence or absence of a link between a node pair. The predictor features are metrics computed from the graph structure or node attributes which describe the given pair. This approach was first introduced by Liben-Nowell and Kleinberg [27], who use graph topological features to study a co-authorship network. The supervised approach is also used in several later works [6, 25, 39].

A key challenge for supervised link prediction is designing an effective set of features for the task. Some works use simple topological features such as the number of common neighbors and the Adamic/Adar index [12], while others use more complex features [11]. For detailed surveys on supervised link prediction methods, please refer to [6, 14, 27].

Our method is also a supervised method based on graph topology. Specifically, we train a classifier on our proposed motif-based features. Our experiments show that this classifier outperforms one trained on a combination of the graph topology features used in the prior work mentioned above. The proposed prediction method intends to improve on local measures by looking into higher-order structures among nodes (i.e., motifs).

Applying supervised methods to link prediction requires preparing a classification dataset for training the model and testing its performance. To ensure that the model will not over-fit to a specific set of samples, a set of existing links must be hidden from the model. If we can observe the graph as it evolves over time, then one approach is to take snapshots of the graph at different times. Edges present in the graph in a snapshot are used as the training set, while edges present in the graph at a later point form the test set. This approach is widely adopted by several studies [27]. If the graph evolution cannot be observed, the edges are usually sampled uniformly at random among the available ones. In either case, a set of negative examples (pairs of unconnected nodes) needs to be sampled from (a snapshot of) the graph. Our method for constructing the classification dataset has some unique features compared to prior work, as we discussed in Section 3.

Feature Learning. The most sophisticated approach to link prediction in the literature is to allow the model to learn by itself which (latent) features are important for the link prediction task. Feature learning methods such as matrix factorization or deep learning depend on the examination of graph topology and structural features to compute score functions based on pairwise similarity metrics.

Approaches based on matrix factorization model the graph as an $N \times N$ matrix in which N represents the number of nodes, and then predict a link by using matrix decomposition. For example, Menon and Elkan [30] consider link prediction as a matrix completion problem and solve it using a non-negative matrix factorization (NMF) method. The basic idea is to let the model learn latent features from the topological structure of a partially observed graph, and then use the model to approximate the unobserved part of the graph via matrix completion. However, NMF performs best when combining the latent features with features based on node attributes. In our work, we use only topological features, so we compare our method to NMF using the latent

(topological) features, and show that our method significantly outperforms NMF.

Deep learning is another very popular form of feature learning. In particular, graph convolutional networks (GCNs) have recently emerged as a powerful tool for representation learning on graphs [21]. GCNs have also been successfully used for link prediction [40, 51]. For example, SEAL [51] is a framework which fits a graph neural network to small subgraphs around the example edges in the dataset. By doing so, it learns latent features in the neighborhood structure of the graph which indicate the presence or absence of a link. Therefore, it is very similar in spirit to our current work. Nevertheless, we show that our motif-based features are able to achieve a much higher prediction accuracy than the one obtained by SEAL.

6. CONCLUSION

We presented a new approach for link prediction in undirected graphs that relies on using the distribution of k -motifs that a pair of nodes appears in to predict whether a link exists between these two nodes. We use $k \in \{3, 4, 5\}$, which distinguishes our approach from prior approaches that rely on topological features: prior approaches are based on common neighbors, while our approach is based on higher-order analysis of the topology. Our approach treats the link prediction problem as a classification problem, so building the classification dataset is a key step. We pointed out two issues related to this step that were not adequately addressed by prior work. First, it is important to treat positive and negative example edges in the same way. Prior approaches achieve this by removing positive example edges from the graph, and we showed that an alternative (and better) way is to insert negative example edges in the graph. Second, when sampling pairs of nodes to find negative example edges, the shortest-path distance between the sampled nodes affects prediction accuracy, with shorter distances increasing the difficulty of the problem. Thus, it is important to control this parameter when building the classification dataset (in our case we used a 50/50 split between distance 2 and 3). Given a classification dataset constructed with these two issues properly addressed, we showed that it is possible, by using off-the-shelf classifiers, to achieve substantial improvement in prediction accuracy compared to prior methods based on topological features or feature learning.

For future work, it would be interesting to explore the use of graphlet-based features instead of motif-based features. Graphlets are induced subgraphs, whereas the motifs that we currently use are partial non-induced subgraphs, and it would be interesting to see if graphlets can provide accuracy gains if used alone or in combination with motifs. Another direction for future work is exploring the temporal aspects of graph evolution. Link prediction is ultimately a temporal problem, where we predict links in the future based on a snapshot of the graph in the present. If every edge has a creation timestamp, temporal motifs can be defined by putting a label on each edge which indicates the relative order of its appearance in the motif. These temporal motif features would capture the graph evolution at a fine resolution, and could perhaps increase the prediction accuracy.

7. REFERENCES

- [1] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.
- [2] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield. Efficient graphlet counting for large networks. In *Proc. IEEE International Conference on Data Mining (ICDM)*, pp. 1–10, 2015.
- [3] L. M. Aiello, A. Barrat, R. Schifanella, C. Cattuto, B. Markines, and F. Menczer. Friendship prediction and homophily in social media. *ACM Transactions on the Web (TWEB)*, 6(2):9, 2012.
- [4] E. M. Airoldi, D. M. Blei, S. E. Fienberg, E. P. Xing, and T. Jaakkola. Mixed membership stochastic block models for relational data with application to protein-protein interactions. In *International Biometrics Society Annual Meeting*, volume 15, 2006.
- [5] M. Al Hasan and M. J. Zaki. A survey of link prediction in social networks. In *Social Network Data Analytics*, pp. 243–275. 2011.
- [6] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006.
- [7] A.-L. Barabási. Scale-free networks: a decade and beyond. *Science*, 325(5939):412–413, 2009.
- [8] M. Bressan, F. Chierichetti, R. Kumar, S. Leucci, and A. Panconesi. Counting graphlets: Space vs. time. In *Proc. ACM Int. Conf. on Web Search and Data Mining (WSDM)*, pp. 557–566, 2017.
- [9] H. Chen, X. Li, and Z. Huang. Link prediction approach to collaborative filtering. In *Proc. Joint Conf. on Digital Libraries (JCDL)*, pp. 141–142, 2005.
- [10] P. M. Chuan, M. Ali, T. D. Khang, N. Dey, et al. Link prediction in co-authorship networks based on hybrid content similarity metric. *Applied Intelligence*, 2017.
- [11] W. Cukierski, B. Hammer, and B. Yang. Graph-based features for supervised link prediction. In *Int. Joint Conf. on Neural Networks (IJCNN)*, 2011.
- [12] M. Fire, L. Tenenboim, O. Lesser, R. Puzis, L. Rokach, and Y. Elovici. Link prediction in social networks using computationally efficient topological features. In *Privacy, Security, Risk and Trust (PASSAT)*, 2011.
- [13] F. Folino and C. Pizzuti. Link prediction approaches for disease networks. In *Proc. Int. Conf. on Information Technology in Bio and Medical Informatics*, 2012.
- [14] F. Gao, K. Musial, C. Cooper, and S. Tsoka. Link prediction methods and their accuracy for different social networks and network metrics. *Scientific Programming*, 2015:1, 2015.
- [15] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.
- [16] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2016.
- [17] E. Hussein, A. Ghanem, V. V. dos Santos Dias, C. H. C. Teixeira, G. AbuOda, M. Serafini, G. Siganos, G. De Francisci Morales, A. Aboulmaga, and M. J. Zaki. Graph data mining with arabesque. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2017.
- [18] K. Juszczyszyn, P. Kazienko, and K. Musiał. Local topology of social network based on motif analysis. In *Proc. Int. Conf. on Knowledge-Based and Intelligent Information and Engineering Systems*, 2008.
- [19] K. Juszczyszyn, K. Musial, and M. Budka. Link prediction based on subgraph evolution in dynamic social networks. In *Proc. IEEE Int. Conf. on Social Computing (SocialCom)*, 2011.
- [20] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.
- [21] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [22] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
- [23] Y. Lavin, R. Batra, and L. Hesselink. Feature comparisons of vector fields using earth mover’s distance. In *Proceedings of the conference on Visualization’98*, pp. 103–109. IEEE Computer Society Press, 1998.
- [24] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 462–470, 2008.
- [25] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proc. Int. Conf. on World Wide Web*, 2010.
- [26] Y. Liang, L. Huang, and Z. Wang. Link prediction in social network based on local information and attributes of nodes. In *Journal of Physics: Conference Series*, volume 887, p. 012043, 2017.
- [27] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.
- [28] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 243–252, 2010.
- [29] L. Lu and T. Zhou. Link prediction in complex networks: A survey. *arXiv preprint arXiv:1010.0725*, 2010.
- [30] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*, 2011.
- [31] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.
- [32] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, 2004.
- [33] M. E. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, 2001.
- [34] T. Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, 35(2):159–167, 2013.
- [35] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814, 2005.
- [36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer,

- R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [37] N. Pržulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.
- [38] J. R. Quinlan. Decision trees as probabilistic classifiers. In *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 31–37. Elsevier, 1987.
- [39] H. R. Sa and R. B. Prudencio. Supervised learning for link prediction in weighted networks. In *Proc. Int. Workshop on Web and Text Intelligence*, 2010.
- [40] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *Proc. European Semantic Web Conf. (ESWC)*, pp. 593–607, 2018.
- [41] D. S. Schneider, K. L. Hudson, T.-Y. Lin, and K. V. Anderson. Dominant and recessive mutations define functional domains of toll, a transmembrane protein required for dorsal-ventral polarity in the drosophila embryo. *Genes and Development*, 5(5):797–807, 1991.
- [42] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature Genetics*, 31(1):64, 2002.
- [43] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu. Scalable proximity estimation and link prediction in on-line social networks. In *Proc. ACM SIGCOMM Conf. on Internet Measurement*, pp. 322–335, 2009.
- [44] E. Soutoglou and I. Talianidis. Coordination of PIC assembly and chromatin remodeling during differentiation-induced gene activation. *Science*, 295(5561):1901–1904, 2002.
- [45] C. H. Teixeira, A. J. Fonseca, M. Serafini, G. Siganos, M. J. Zaki, and A. Aboulmaga. Arabesque: a system for distributed graph mining. In *Proc. Symp. on Operating Systems Principles (SOSP)*, pp. 425–440, 2015.
- [46] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller. VERSE: Versatile graph embeddings from similarity measures. In *Proc. Web Conf. (WWW)*, 2018.
- [47] J. Ugander, L. Backstrom, and J. Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proc. Int. Conf. on World Wide Web (WWW)*, pp. 1307–1318, 2013.
- [48] A. Vazquez, R. Dobrin, D. Sergi, J.-P. Eckmann, Z. Oltvai, and A.-L. Barabási. The topological relationship between the large-scale attributes and local interaction patterns of complex networks. *Proceedings of the National Academy of Sciences*, 101(52):17940–17945, 2004.
- [49] P. Wang, B. Xu, Y. Wu, and X. Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.
- [50] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015.
- [51] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 5171–5181, 2018.