Link Prediction via Higher-Order Motif Features

Ghadeer Abuoda[1], Gianmarco De Francisci Morales[2], and Ashraf Aboulnaga[3]

[1] College of Science and Engineering, HBKU, Doha, Qatar. gabuoda@hbku.edu.qa
[2] ISI Foundation, Turin, Italy. gdfm@acm.org
[3] Qatar Computing Research Institute, Doha, Qatar. aaboulnaga@hbku.edu.qa

Abstract. Link prediction requires predicting which new links are likely to appear in a graph. In this paper, we present an approach for link prediction that relies on higher-order analysis of the graph topology, well beyond the typical approach which relies on 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. In addition, we propose two optimizations to construct the classification dataset from the graph. First, we propose adding negative examples to the graph as an alternative to the common approach of removing 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 distance. We experimentally demonstrate that using our proposed motif features in off-the-shelf classifiers results in up to 10 percentage points increase in accuracy over prior topology-based and feature-learning methods.

Keywords: Link prediction · Motifs.

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 $t_2 > t_1$ [24]. 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 [4]. Link prediction has also been used to detect credit card fraud in the cybersecurity domain [24], to predict protein-protein interactions in bioinformatics [5, 17], for shopping and movie recommendation in e-commerce [10], 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) [7]. Most topological features are based on common neighbors, i.e., they rely on the idea of 'closing triangles' [30]. More advanced approaches such as non-negative matrix factorization (NMF) and graph embeddings have also been tried recently [27, 42]. However, traditional topological features that rely on common neighbors, such as the Jaccard index and Adamic/Adar measure [2], have proven to be very strong baselines which are hard to beat [42].

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 [3, 41]. Given the outstanding results of traditional topological features, it is natural to look towards more complex features based on motifs for better predictive power [17, 32, 34]. In this paper, we show that using features based on higher-order motifs with a carefully designed classification dataset significantly improves 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].

When extracting features, two technical 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. In our experiments, this variant consistently outperforms the former in terms of accuracy. For the second issue, we show that distance between nodes in negative examples is an important factor that should be controlled for when creating a dataset (an under-appreciated fact in the link prediction literature [44]).

The main contributions of this study are as follows:

- We show that 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 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 find which new edges are likely to appear at time $t_2 > t_1$, i.e., to predict the set $\{e : e \notin E_{t_1} \wedge e \in E_{t_2}\}$. We assume G is undirected and unweighted, and the set of nodes V does not change in time.

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 a static graph, and the label $L(u, v) = 1$ if $(u, v) \in E$, else $L(u, v) = -1$.

The edges in the graph can be used as positive examples, while for negative examples we can sample pairs of nodes in the graph which are not connected by an edge. We call these pairs of nodes *negative edges*.

2.1 Motifs

Motifs are small, connected, non-isomorphic subgraphs which appear in a larger graph [29, 38]. Each k-motif represents a topological pattern of interconnection between k nodes in a graph. We denote each motif as '$\mathrm{m}k.n$' 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 (a list of motifs of sizes 3–5 is available in the extended version of this paper [1]).

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 [28]. Therefore, the usefulness of motifs to capture the macro structure of a graph is well established [43]. However, for our purpose, we are more interested in their ability to capture the micro structure of the graph (i.e., the neighborhood).

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 [9, 41]. 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

The features in our model correspond to the number of occurrences of an edge (positive or negative) 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 $k \in \{3, 4, 5\}$. 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.

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. Positive edges will have a mix of both features, but negative edges will never appear in a closed triangle, by construction. Thus, 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,

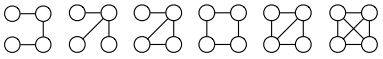
			Features					
								
Graph	Edge	Class	m4.1	m4.2	m4.3	m4.4	m4.5	m4.6
	1-2	Positive	1	0	0	0	0	0
	2-3	Negative	1	0	0	0	0	0

Fig. 1: Motif features when positive examples are removed from the graph (`RMV`).

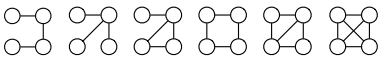
			Features					
								
Graph	Edge	Class	m4.1	m4.2	m4.3	m4.4	m4.5	m4.6
	1-2	Positive	2	2	4	0	1	0
	2-3	Negative	4	1	3	1	1	0

Fig. 2: Motif features when negative examples are inserted into the graph (`INS`).

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 already in the graph?"

In the first method, `RMV`, we remove the example positive edges from the graph and extract the features by looking at motifs that contain both endpoints of a removed edge. 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 1. Let green edges be positive examples, red edges be negative examples, and black edges be part of the graph but not in the classification dataset (i.e., not 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 sampled but not inserted. Removing edge $(1, 2)$ changes the motifs in this neighborhood. For example, motifs m4.2 and m4.3 do not appear even though edge $(1, 2)$ was part of instances of these motifs in the original unmodified graph. After removing edge $(1, 2)$, the only 4-motif that appears is m4.1, which appears once. Since it contains the nodes in edges $(1, 2)$ and $(2, 3)$, both edges have a value 1 for feature m4.1.

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. All motifs can appear as features, and an example edge contributes to all the motifs it is part of. Figure 2 shows the same example as Figure 1, but now the negative edge $(2, 3)$ is added to the graph. Each feature of an example edge (positive or negative) corresponds to a motif which includes the edge itself. As an illustration of extracting motif features, consider m4.2 and m4.3 in Figure 2. Motif m4.2 occurs twice in the graph, $(0, 1)$-$(1, 2)$-$(1, 3)$ and $(0, 2)$-$(1, 2)$-$(2, 3)$. Both occurrences contain edge $(1, 2)$ while only one contains edge $(2, 3)$, so edge $(1, 2)$ has a value 2 for feature m4.2 while edge $(2, 3)$ has a value 1. There are four occurrences of motif m4.3 in the graph, obtained by removing one of the edges $(0, 1)$, $(1, 3)$, $(0, 2)$, or $(2, 3)$. All of these occurrences include edge $(1, 2)$ but only three include edge $(2, 3)$, so edge $(1, 2)$ has a value 4 for feature m4.3 while edge $(2, 3)$ has a value 3.

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 2. That is, the extracted motifs are not fully independent of the sampling. While this is not desirable, we verify that the occurrence of these cases in practice is very rare, so they do not affect the learning process in any significant way.

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, given the assumption that no edge is easier 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 for 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 short distances ($d \in \{2, 3\}$), as these represents the hardest cases. 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 imbalance. In a typical graph, most pairs of nodes do not have an edge 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.

Graph	|V|	|E|	Avg. Deg.	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

(a) Basic statistics about the graph datasets.

Graph	# Pos. Edges	# Neg. Edges
Amazon	20 000	20 000
CondMat	2000	2000
AstroPh	5000	5000

(b) No. of positive and negative edges sampled for the classification datasets.

Table 1: Statistics for the graphs and the classification datasets.

4 Experimental Evaluation

4.1 Experimental Setup

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 1a 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. Nodes are products, and an edge between any two nodes shows that the two products have been frequently bought together. The second dataset, CondMat, 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. 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.

Experimental Settings. For each graph, we extract a classification dataset for which we compute 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 1b shows the number of examples chosen from each graph. We extract motif features for example edges by using the Arabesque parallel graph mining framework [41]. We then group by motif, count the occurrences, and finally normalize the counts to create a feature vector which represents the motif distribution of the neighborhood of the edge.[2]

To train the classification models we use the scikit-learn Python library.[3] We train naïve Bayes (NB), logistic regression (LR), decision tree (DT), k-nearest neighbor (KNN), gradient boosted decision tree (GB), and random forest (RF) models. All performance results are computed via 10-fold cross-validation.

Baselines. We use two types of baselines. The first type includes traditional topological features such as triangle closure and paths. We compare our features against common neighbors, Jaccard coefficient, Adamic/Adar measure, Preferential Attachment, rooted PageRank, and Katz index. Of these methods, PageRank and Katz benefit from inserting negative edges in the graph, so we use `INS` with these two methods.

[1] `http://konect.uni-koblenz.de`

[2] Code available at `https://github.com/GhadeerAbuoda/LinkPrediction`.

[3] `http://scikit-learn.org`

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

`RMV`

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

`INS`

Table 2: Classification performance on Amazon for `RMV` vs. `INS`.

The second type of baseline includes more complex techniques such as matrix decomposition 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 (`RMV`), as commonly done in the literature [27]. 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 [45]. SEAL uses subgraph extraction around the example edge to extract latent features, learned via a neural network. This framework has experimentally outperformed other existing deep learning methods such as node2vec and LINE [16, 40].

Evaluation Metrics. We evaluate the algorithms via the following 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 from the scores produced by the classifiers. It represents the probability that a classifier will rank a random positive example higher than a random 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, $FPR = \frac{FP}{FP+TN}$. This measure is useful to understand the effect of graph distance of the negative examples. Better classifiers obtain lower FPR.

4.2 Removing Positive Edges vs. Inserting Negative Edges

Table 2 shows 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 $k = 3$, size $k = 4$, and size $k = 5$. The last column shows 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, it is clear that `INS` consistently has higher accuracy than `RMV`. The difference grows smaller as we add

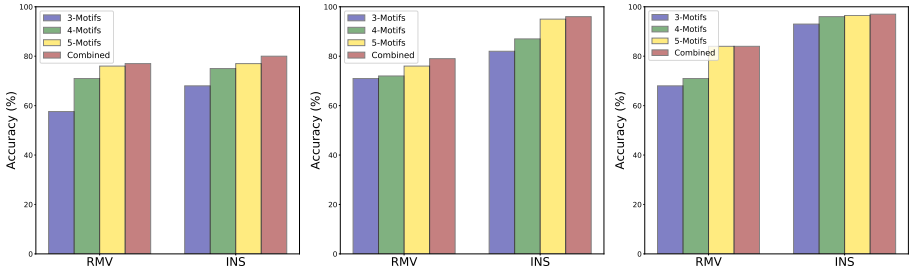


Fig. 3: Classification accuracy of a Random Forest (RF) classifier when using different motif features and different feature extraction methods (`RMV` vs. `INS`).

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 2 also reports 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.

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

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. We return to the reason why the `INS` feature extraction method is superior in Section 4.4.

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 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 supplement this information to the 5-motif features.

Finally, Table 3 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.

Graph	RMV			INS		
	ACC (%)	AUC	FPR	ACC (%)	AUC	FPR
Amazon	77.0	0.77	0.27	79.6	0.80	0.18
CondMat	79.0	0.79	0.04	96.0	0.96	0.04
AstroPh	84.0	0.84	0.30	96.5	0.97	0.02

Table 3: Classification performance of a Random Forest (RF) classifier with combined motif features and different feature extraction methods (RMV vs. INS).

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**

Table 4: Accuracy of the RF classifier (%) with combined motif features (INS) vs. baseline classifiers.

4.3 Comparison with Baselines

To compare against the baseline topological features proposed in prior work, we train a RF on each of these features, and one on the combination of all of the features. The upper part of Table 4 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. The next two rows show path-dependent topological features. For rooted 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 (for $\beta = 0.1$). The accuracy of the topological features is in the range 55–85%. 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 4 can be viewed as the best possible accuracy with current state-of-the-art topological features. However, the motif features achieve much higher accuracy. Specifically, they are 7 to 10 percentage points better in accuracy, which is significant given that advanced features extracted via graph embeddings and deep learning reportedly struggle to beat the traditional topological features [42].

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 [27] is not very competitive, as shown in Table 4. 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, which is quite disappointing. Moreover, adding the NMF features to the topological ones does not improve accuracy by much (only the AstroPh dataset sees some improvement).

Finally, we compare our model with SEAL [45], a recent link prediction framework which uses deep learning (graph neural networks). We test the framework with its default hyperparameters. Interestingly, SEAL only achieves around 70% accuracy on Amazon and 80% accuracy on the other two datasets. SEAL learns on one- or two-hop subgraphs extracted around the tested edge, which is somewhat equivalent to looking into common neighbors. However, the accuracy achieved by SEAL is lower than with the combined topological features.

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 does not improve on average, and actually drops 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., the length of the random walks may not be appropriately tuned).

Thus, the overall takeaway from Table 4 is that RF with motif features is more accurate than all the 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 4 shows the relative importance of the features as inferred by 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. The most predictive feature is always a 5-motif one, which is another indication of the predictive power of deeper structural features. However, it changes from dataset to dataset, and might be domain specific.

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 two 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.

4.4 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 distance between the motif distribution for $k = 3$ and $k = 4$ in the original graph, and the ones obtained by `RMV` and `INS`. We use two different distance functions to perform the comparison: Earth Mover's Distance (EMD), and Kullback-Leibler Divergence (KLD). Table 5 reports the results. If our hypothesis is correct, then

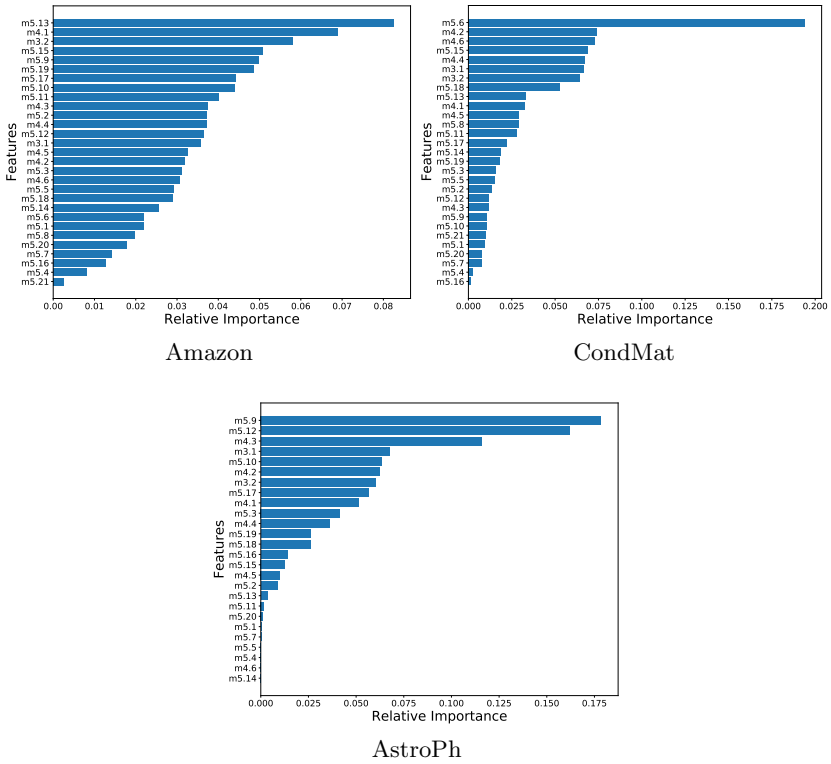


Fig. 4: Feature importance across the three datasets, as inferred from the Random Forest model. In all three datasets the most important feature is a 5-motif, however the specific motif varies by dataset.

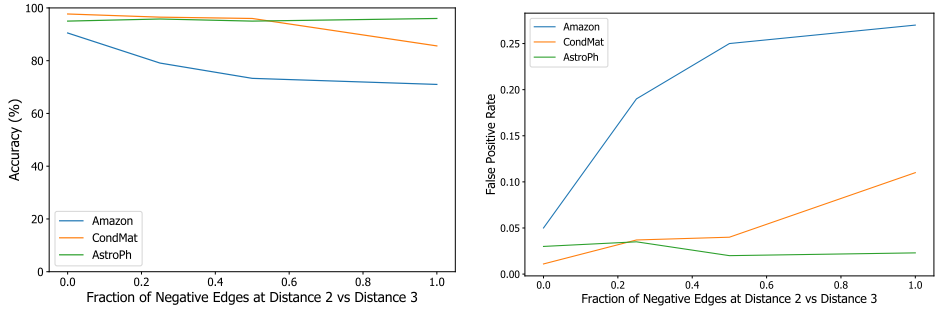
`INS` should have a smaller distance than `RMV`. This is indeed the case for two out of three graphs, for both distance functions, which gives us confidence that our hypothesis is a step in the right direction. However, AstroPh behaves differently, with `RMV` having a smaller distance than `INS`. Therefore, we cannot draw a definitive conclusion, and further study is necessary to fully understand the difference between these two feature extraction methods.

4.5 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.

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

Table 5: Earth Mover's Distance (EMD) and KL 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 feature extraction method is more faithful to the original graph.



(a) (b)

Fig. 5: Classification accuracy and false positive rate as a function of the fraction of negative examples at distance 2 (vs. distance 3).

Figure 5a 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 5b, 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.

5 Related work

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

Graph Motifs. Motifs are patterns of connectivity that occur significantly more frequently in the given graph than expected by chance alone [43]. Graph motifs have numerous applications, for example, they have been used to classify graphs into "superfamilies" [37], and they have been used in combination with machine learning to determine the most appropriate model for a given real-world network [39]. Palla et al. [31] also show that 4-cliques reveal community structure in word associations and protein-protein interaction graphs. In several social me-

dia analysis studies [18, 19], 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. 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.

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 (including motifs).

In most unsupervised methods, a heuristic is 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, 26]. The heuristic is typically a similarity measure, and can be based on application-specific node attributes or on the graph topology.

While node attributes can achieve a high degree of accuracy, 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 [30]. More advanced features include some form of regularization of the count, such as the Jaccard coefficient of the two sets of neighbors, the Adamic/Adar index [2], which discounts the contribution of high-degree nodes, and preferential attachment [8], which gives a higher likelihood to links between high degree vertices.

Conversely, path-based features look at the global graph structure. A representative path-based feature is the 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 references [15] and [25]. Several studies indicate that unsupervised methods are fundamentally unable to cope with dynamics, imbalance, and other complexities of real-world graphs [6, 25]. However, similarity indices can easily be used by supervised methods as features for a machine learning model.

In supervised methods, link prediction is usually cast as a binary classification problem. The 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 [6, 23, 24, 35].

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, 24].

Some prior work has used motif-like features for link prediction problems. For example, Hulovatyy et al. [17] use features extracted from graphlets for link prediction. Theirs is an unsupervised method that uses a different type of feature extraction compared to our approach. Graphlet-based features are also used in [32] for link prediction. However, the focus of that paper is link prediction in temporally evolving graphs, while we focus on static graphs.

A more sophisticated approach to link prediction 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, graph embedding, or deep learning examine the graph topology to learn a representation that can be used in machine learning tasks.

Matrix factorization models the graph as an $N \times N$ matrix, and then predicts a link by using matrix decomposition. For example, Menon and Elkan [27] 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. Higher-order network embeddings [33, 34] use a motif-based matrix formulation to learn a representation of the graph that can be used for link prediction.

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]. Lee et al. [22] propose a GCN technique that uses motif information to improve accuracy in classification tasks. GCNs have also been successfully used for link prediction [36, 45]. For example, SEAL [45] 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 the current work. In this paper, we compare with NMF and SEAL as examples from the class of representation learning techniques, and we show that our method outperforms these more complex methods.

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 pointed out two issues related to the task 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 for this parameter when building the classification dataset. Finally, we showed that, by using off-the-shelf classifiers, our motif features achieve substantial improvement in prediction accuracy compared to prior methods based on topological features or feature learning.

Bibliography

[1] Abuoda, G., De Francisci Morales, G., Aboulnaga, A.: Link prediction via higher-order motif features. arXiv preprint arXiv:1902.06679 (2019)
[2] Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social Networks 25(3), 211–230 (2003)
[3] Ahmed, N.K., Neville, J., Rossi, R.A., Duffield, N.: Efficient graphlet counting for large networks. In: ICDM. pp. 1–10 (2015)
[4] Aiello, L.M., Barrat, A., Schifanella, R., Cattuto, C., Markines, B., Menczer, F.: Friendship prediction and homophily in social media. TWEB 6(2), 9 (2012)
[5] Airoldi, E.M., Blei, D.M., Fienberg, S.E., Xing, E.P., Jaakkola, T.: Mixed membership stochastic block models for relational data with application to protein-protein interactions. In: International Biometrics Society Annual Meeting. vol. 15 (2006)
[6] Al Hasan, M., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: Workshop on Link Analysis, Counter-terrorism and Security (2006)
[7] Al Hasan, M., Zaki, M.J.: A survey of link prediction in social networks. In: Social Network Data Analytics, pp. 243–275 (2011)
[8] Barabási, A.L.: Scale-free networks: a decade and beyond. Science 325(5939), 412–413 (2009)
[9] Bressan, M., Chierichetti, F., Kumar, R., Leucci, S., Panconesi, A.: Counting graphlets: Space vs. time. In: WSDM. pp. 557–566 (2017)
[10] Chen, H., Li, X., Huang, Z.: Link prediction approach to collaborative filtering. In: JCDL. pp. 141–142 (2005)
[11] Cukierski, W., Hamner, B., Yang, B.: Graph-based features for supervised link prediction. In: IJCNN (2011)
[12] Fire, M., Tenenboim, L., Lesser, O., Puzis, R., Rokach, L., Elovici, Y.: Link prediction in social networks using computationally efficient topological features. In: Proc. Int. Conf. on Privacy, Security, Risk and Trust (PASSAT) (2011)
[13] Folino, F., Pizzuti, C.: Link prediction approaches for disease networks. In: Proc. Int. Conf. on Information Technology in Bio and Medical Informatics (2012)
[14] Gao, F., Musial, K., Cooper, C., Tsoka, S.: Link prediction methods and their accuracy for different social networks and network metrics. Scientific Programming 2015, 1 (2015)
[15] Getoor, L., Diehl, C.P.: Link mining: a survey. SIGKDD Explorations Newsletter 7(2), 3–12 (2005)
[16] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD (2016)
[17] Hulovatyy, Y., Solava, R.W., Milenković, T.: Revealing missing parts of the interactome via link prediction. PLOS ONE (2014)
[18] Juszczyszyn, K., Kazienko, P., Musiał, K.: Local topology of social network based on motif analysis. In: Proc. Int. Conf. on Knowledge-Based and Intelligent Information and Engineering Systems (2008)
[19] Juszczyszyn, K., Musial, K., Budka, M.: Link prediction based on subgraph evolution in dynamic social networks. In: SocialCom (2011)
[20] Katz, L.: A new status index derived from sociometric analysis. Psychometrika 18(1), 39–43 (1953)
[21] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
[22] Lee, J.B., Rossi, R.A., Kong, X., Kim, S., Koh, E., Rao, A.: Higher-order graph convolutional networks. arXiv preprint arXiv:1809.07697 (2018)
[23] Leskovec, J., Huttenlocher, D., Kleinberg, J.: Predicting positive and negative links in online social networks. In: WWW (2010)
[24] Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. of the Assoc. for Information Science and Technology 58(7), 1019–1031 (2007)

[25] Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: KDD. pp. 243–252 (2010)
[26] Lu, L., Zhou, T.: Link prediction in complex networks: A survey. arXiv preprint arXiv:1010.0725 (2010)
[27] Menon, A.K., Elkan, C.: Link prediction via matrix factorization. In: ECML-PKDD (2011)
[28] Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., Alon, U.: Superfamilies of evolved and designed networks. Science 303(5663), 1538–1542 (2004)
[29] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science 298(5594), 824–827 (2002)
[30] Newman, M.E.: Clustering and preferential attachment in growing networks. Physical Review E 64(2), 025102 (2001)
[31] Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043), 814 (2005)
[32] Rahman, M., Al Hasan, M.: Link prediction in dynamic networks using graphlet. In: ECML-PKDD. pp. 394–409 (2016)
[33] Rossi, R.A., Ahmed, N.K., Koh, E.: Higher-order network representation learning. In: Companion Proc. of the Web Conf. (WWW). pp. 3–4 (2018)
[34] Rossi, R.A., Ahmed, N.K., Koh, E., Kim, S., Rao, A., Yadkori, Y.A.: HONE: Higher-order network embeddings. arXiv preprint arXiv:1801.09303 (2018)
[35] Sa, H.R., Prudencio, R.B.: Supervised learning for link prediction in weighted networks. In: Proc. Int. Workshop on Web and Text Intelligence (2010)
[36] Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC. pp. 593–607 (2018)
[37] Schneider, D.S., Hudson, K.L., Lin, T.Y., Anderson, K.V.: 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)
[38] Shen-Orr, S.S., Milo, R., Mangan, S., Alon, U.: Network motifs in the transcriptional regulation network of Escherichia coli. Nature Genetics 31(1), 64 (2002)
[39] Soutoglou, E., Talianidis, I.: Coordination of PIC assembly and chromatin remodeling during differentiation-induced gene activation. Science 295(5561), 1901–1904 (2002)
[40] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: Large-scale information network embedding. In: WWW. pp. 1067–1077 (2015)
[41] Teixeira, C.H., Fonseca, A.J., Serafini, M., Siganos, G., Zaki, M.J., Aboulnaga, A.: Arabesque: A system for distributed graph mining. In: SOSP. pp. 425–440 (2015)
[42] Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: VERSE: Versatile graph embeddings from similarity measures. In: WWW (2018)
[43] Vazquez, A., Dobrin, R., Sergi, D., Eckmann, J.P., Oltvai, Z., Barabási, A.L.: The topological relationship between the large-scale attributes and local interaction patterns of complex networks. Proc. Natl. Academy of Sciences 101(52), 17940–17945 (2004)
[44] Yang, Y., Lichtenwalter, R.N., Chawla, N.V.: Evaluating link prediction methods. Knowledge and Information Systems 45(3), 751–782 (2015)
[45] Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: NeurIPS. pp. 5171–5181 (2018)