COMP4801 Final Year Project 2017-18

Final Report

Project Title:	**Discovering and Querying Meta-Graphs in Large Heterogeneous Information Networks**
Supervisor:	**Dr. Reynold C.K. Cheng**
Student Name:	**Fung Yuet**
UID:	**-**
FYP account:	**fyp17045**
Date of Submission:	**15th April, 2018**

1 Abstract

Heterogeneous Information Network (HIN) is a graphical representation of a dataset. This depiction enables researchers to carry out unprecedented knowledge discovery tasks and to discover unnoticeable but promising facts from the data. Meta-paths and meta-graphs are two meta-level data structures. They were proposed to aid the task of similarity searches and others. However, discovering those structures is difficult manually. Although there has been some effort put into the study of meta-paths generation, most of them cannot scale well enough to handle meta-graphs discoveries. This work therefore investigates the problem of how to discover meta-graphs in large HINs efficiently. A greedy algorithm adapted from an existing framework was proposed in this work to select the most relevant meta-graph gradually and to include it in a regression model. Extensive empirical studies show that the proposed framework discovers essential meta-graphs forming an accurate prediction model effectively and efficiently.

Table of Contents

List of Figures

Abbreviations

HIN	Heterogeneous Information Network
FSPG	Forward Stagewise Path Generation
FSSG	Forward Stagewise Structure Generation
LCA	Lowest Common Ancestor
BSCSE	Biased Structure Constrained Subgraph Expansion
MLAR	Revised model of Least-Angle Regression Model
AMPG	Automatic meta-path generation
SMPG	Set-expansion meta-path generation
ROC	Receiver Operating Characteristics
AUC	Area under curve

1 Introduction

Background information of the topic is presented in Chapter 1.1. In Chapter 1.2, there is a discussion of some prior works on the related research topics while the problem statement and objectives of this project are introduced in Chapter 1.3 at the end of Chapter 1.

1.1 Background

Data has been increasingly available as the use of the Web and social media by people has been surging. This copious amount of rich information enables the study of knowledge discovery in data aimed for better understanding of the human behaviour. For instance, researchers mine patterns from the training data for the use of trend projection; researchers investigate the topic of relevance between objects to enhance the similarity search. In recent decades, mining in information networks has aroused growing attention from researchers because many datasets can be organized into a graph whose complex structural characteristics allows one to dig out promising knowledge. An information network is a graph *G = (V, E)*, where *V* is the set of nodes (i.e. *objects*) and *E* is the set of edges (i.e. *relationships*) [17]. An example is depicted in Figure 1(a). For each node, it has one or more associated node classes; for each edge, it has one associated edge type. Thus, there are two more important functions: τ: $V \rightarrow \mathrm{A}$; φ: $E \rightarrow \mathcal{R}$. The first function τ is the node class matching function where each object $v \in V$ matches to one or more node classes $\tau(v) \in \mathrm{A}$. Likewise, the second function φ is the edge type matching function where each edge $e \in E$ belongs to one edge type $\varphi(e) \in \mathcal{R}$. Accordingly, there are two kinds of information network:

if $|\mathrm{A}| = 1$ and $|\mathcal{R}| = 1$, it is a homogeneous information network

if $|\mathrm{A}| > 1$ or $|\mathcal{R}| > 1$, it is a heterogeneous information network (*aka HIN*) [14]

Homogeneous information network mining has been studied since the last few decades during which researchers have been developing methods for analysing homogeneous information networks on the task of clustering, ranking and link prediction [14]. It although seems feasible to extend some of these techniques to handle the study of HINs, most of them cannot be directly applied to the problem. It is because the schema of an HIN is far more complicated than the one in a homogeneous information network and this enables an HIN to express richer information. In addition, node classes and edge types are different across objects and relations. Consequently, considering them as identical as those in the case of homogeneous information network loses the semantic meaning and possibly the valuable information one

would have mined from the network. Therefore, researchers should develop a new set of methodologies and principles in studying HINs.

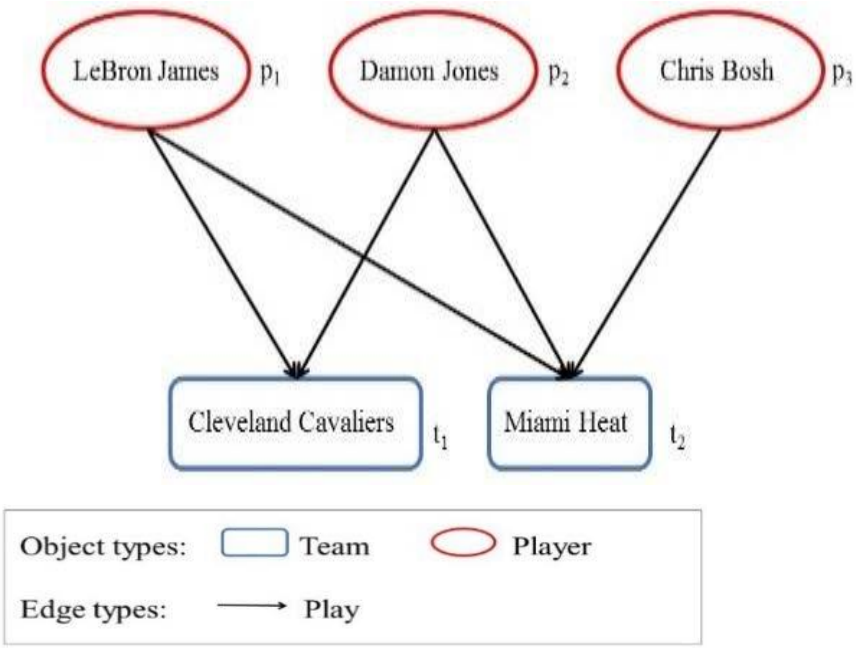


Figure 1(a) HIN

One interesting research topic in HINs analysis is relevance and similarity search. Relevance of two objects defines how similar they are or how tightly they are being related. These research results provoke the analysis of similarity search, classification, clustering and link prediction in HINs [14]. While many works have been done on relevance study, e.g. *Personalized PageRank* [5], *Jaccard's coefficient* [11] and *SimRank* [6], these measures neither consider the different semantic meanings of node classes nor that of edge types. Regarding this issue, Sun. et al [17] proposed the concept of *meta path* and a family of *meta path-based similarity measures.* A meta path is a path comprising of node classes and edge types instead of the actual objects and relations. Node classes and edge types are called meta data in network analysis and thus a path of meta data is recognized as a meta path. Two examples are illustrated in Figure 1(b).

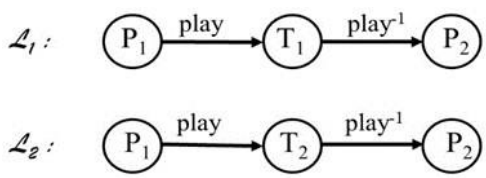


Figure 1(b) Meta paths

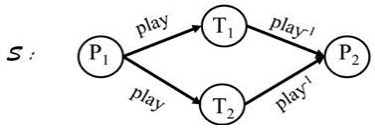


Figure 1(c) Meta-graph

Although meta path has been proved to be useful in many applications, it can only capture simple relationships between the source node and the target one. More importantly, it simplifies all the relations as a single path relation and overlooks the semantic different between a combination of meta paths and a structural relation. Huang et al. [4] subsequently proposed a concept of *meta structure* aiming to provide a data structure depicting complex relations; Fang et al. [2] at the mean time suggested *metagraph*, being a generalisation of meta path and meta structure, to represent various semantic relations. An example of metagraph is shown in Figure 1(c). Meta-graphs (*i.e. meta path, meta structures and metagraph*) have then been proved to be beneficial to product recommendation, community detection, classification and clustering as well as link prediction [1-4,7-8,10].

1.2 Previous works

Despite the substantial studies of meta-graphs application, most of the works assumed these structures are given by experts or are discovered using enumeration or Breadth First Search. Sun et al. [15] assumed the meta paths defining the co-author relationship are given by experts in the study of applying meta paths in prediction and recommendation. Yu et al. [18] and Li et al. [10] both presumed the meta paths will be given in the analysis of recommendation and clustering. Although some researchers attempted to use Breadth First Search to discover the meta path [8,16], they required the user to input the maximum meta path length restricting the search space. Similarly, researchers expected *meta-graph* are given by domain experts [19] or by enumeration [7] for the study of its applications. Since meta-graphs are difficult to define for a complex schema and it is a labour-demanding task [12], while Breadth First Search and enumeration are not efficient for large HINs, it is obvious that researchers should develop an efficient approach to discover these meta-graphs automatically in large heterogeneous information networks.

Some researchers have studied this problem in the last few years attempting to automatically discover and to locate these illustrative data structures in large HINs. Regarding the discovery of meta paths, there are in general two approaches: *example-based training* and *adapted sequential pattern mining*. Example-based training means the user must first provide

positive example pairs. The algorithm framework trains a model based on these pairs and transverses the HIN to discover the meta paths highly explaining the relationship of the given pairs. There are many proposed algorithms using this framework, e.g. FSPG [12], AMPG [1] and SMPG [20], though they differ subtly in some areas. For example, the definitions of the priority score for heuristics pruning and the method of discarding unimportant meta paths. Additionally, Shi et al. [13] proposed a method adapted from well-known knowledge discovery techniques − sequential pattern mining, aiming to simulate mining interesting meta paths as sequential pattern mining. "Generate-and-Discard" suggested by Shi [13] targeted to generate meta paths linking the two target nodes by considering the linkage between the siblings of the two target nodes.

Regarding the discovery of meta-graphs, Fang et al. [2] proposed a heuristic approach to mine meta-graphs from HINs. They suggested to use a set of seed candidate meta-graphs, assuming that two structurally similar meta-graphs are functionally similar. By maximizing the structural similarity of any potential meta-graphs to any seed meta-graphs, it means that the chosen one is structurally and functionally similar to any seed meta-graphs. However, it is unclear whether functional similarity and structural similarity are highly correlated. Moreover, using a set of seed meta-graphs would limit the diversity of candidates. It thus needs more justifications. In short, researchers should develop a more unified and efficient framework in handling automatic discovery of meta-graphs for large heterogeneous information networks.

1.3 Problem statement and Objectives

The goal of this work is to develop a systematic and methodical algorithmic framework allowing efficient discovery of meta-graphs in large heterogeneous information networks. In this work, the method proposed in [12] is adapted with modifications for optimization in the discovery. Below are the definitions of the models used in this study and the specific objectives of this work.

1.3.1 Problem Definitions

DEFINITION 1 (**HETEROGENEOUS INFORMATION NETWORK**).

A heterogeneous information network is a graph G containing V, which is the set of nodes (i.e. objects), and E, which is the set of edges (i.e. relationships). There are two more important functions: $\tau: V \rightarrow \mathrm{A}$*;* $\varphi: E \rightarrow \mathcal{R}$*. The first function* τ *is the node class matching function where each object* $v \in V$ *matches to one or more node classes* $\tau(v) \in \mathrm{A}$*. Likewise, the second function* φ *is the edge type matching function where each edge* $e \in E$ *belongs to one edge type*

$\varphi(e) \in \mathcal{R}$. *Note that* $|\mathrm{A}| > 1$ or $|\mathcal{R}| > 1$.

DEFINITION 2 (**META-GRAPHS**).

Given an HIN G = (V, E, A, $\mathcal{R}$*) with* τ and φ*, a meta-graph is a graph* $\bar{G} = (\bar{V}, \bar{E})$ *where* $\bar{V} \in A$ *and* $\bar{E} \in \mathcal{R}$.

PROBLEM 1 (**RELEVANT META-GRAPHS**).

Given an HIN G = (V, E, A, $\mathcal{R}$*) with* τ and φ*, together with a set of n example pairs* $S_{ep} = \{(s_i,t_i) | i \in [1, \mathrm{n}]\}$ *and a similarity function* $\sigma(s, t|S_{\mathrm{mg}})$*, discover a set of m meta-graphs* $S_{mg} = \{g_i \mid i \in [1, m]\}$ *that capture the characteristics of each pair in* S_{ep}.

1.3.2 Objectives

In this project, studies are separated into three phases and their corresponding objectives are listed as follows:

- Phase One: Use the existing algorithm framework in [12] to implement a program such that *link-only meta-paths* can be discovered.
- Phase Two: Extend and modify the work in Phase One such that *link-only meta-graphs* can be generated and implement the Lowest Common Ancestor (LCA) lookup.
- Phase Three: Conduct analysis and experiments to understand the performance of the proposed algorithm.

The remainder of this report is arranged as follows. The theoretical information of FSSG is introduced in Chapter 2.1.1. There is a detailed illustration of how the algorithm works for mining meta-graphs in Chapter 2.1.2. Chapter 3 delivers a thorough discussion of the experiments and results obtained from extensive empirical studies on the implementation. Chapter 3.3 is primarily dedicated to the analysis of the results of various experimental tasks. Chapter 4 focuses the discussion on the theoretical and technical difficulties encountered. The conclusion of this study is presented in Chapter 5 with a brief discussion on the possible future work.

2 Methodology

The proposed algorithm – Forward Stagewise Structure Generation (FSSG) is introduced in this chapter. There are separate analyses on the theoretical information and the algorithm logic.

2.1 Forward Stagewise Structure Generation (FSSG)

In this chapter, there is an in-depth discussion on the theoretical information of FSSG in Chapter 2.1.1. To better illustrate the idea and the flow of logic of FSSG, an example is shown in Chapter 2.1.2 to walk readers through the execution of FSSG.

2.1.1 Theoretical background

FSSG is an example-based training algorithm whose framework was adapted from FSPG proposed by Meng et al. [12]. FSSG incorporates a two-phase framework in discovering metagraphs in large HINs given a set of positive example pairs. In the first phase of FSSG, it generates meta-graphs that have edge types only leaving the node classes empty. Then FSSG fills in the node classes by using the node class hierarchy. A node class hierarchy shows the relationship between various classes. Figure 2 illustrates an instance of a node class hierarchy. In general, FSSG uses greedy strategies to select a set of meta-graphs with high correlation values which denotes their ability in explaining the relationship of every example pairs under a specific similarity measure.

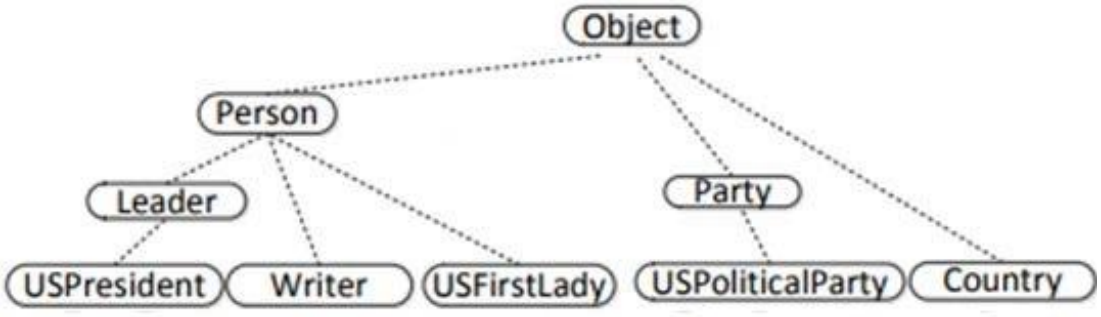


Figure 2. A node class hierarchy [12]

Before executing FSSG, users must provide the program with a set of positive example pairs. These example pairs must be in the form of *(i,j)* where i is the source node and j is the destination node. Given a pair of (*a*,*b*), it means that the user wants to discover meta-graphs that can explain the relationship of *a* and *b*. Given a set of these example pairs, the user would like to locate a collection of meta-graphs that explains the relationship of each pair. For example,

if the user provides two pairs: (*B. Obama*, *M. Obama*) and (*G.W. Bush*, *L. Bush*), the user may want to determine meta-graphs that can explain the relationship of the US President and the US First Lady.

Then, the algorithm generates the same number of *negative* example pairs. It is necessary to include negative example pairs in the discovery process because negative pairs can lower the priority score (i.e. uniqueness) of popular meta-graphs. And thus, the results favour unique meta-graphs instead of common meta-graphs. Next, the algorithm initializes a data structure called **GreedyTree** with a root node containing all the source nodes, their similarity score and the priority score. Here is a short discussion of the two scores. First, a similarity score function determines how close two entities *s* and *t* are related given a set of meta-graphs. Applying this function allows the algorithm to select the best set of meta-graphs in explaining all the example pairs. Below is the equation of similarity function σ.

$$\sigma(g, i\,|S, t) = \frac{1}{|\rho(g, i|S, G)|^{\alpha}} \sum_{x \in \rho(g,i|S,G)} \sigma(x, i+1|\, S, t)$$

$$\sigma(g, n\,|S, t) = 1 \; if \; \chi(n_d) = t$$

where *S* is a meta-graph, *t* is the sink object, $\rho(g, i|S, G)$ is the set of instances at *(i+1)*-th layer. The similarity score n_s and n_t given a meta-graph *S* is $\sigma(n_s, 1 \mid S, n_t)$. This is the definition of BSCSE function proposed by Huang in his work on meta-structure [4]. Second, the priority score is required to perform a heuristic discovery on the **GreedyTree** and it in fact is an upper bound of the actual correlation value. The true correlation function is defined as the standard cosine function as below:

$$\cos(\boldsymbol{m}, \boldsymbol{r}) = \frac{\boldsymbol{m} \cdot \boldsymbol{r}}{\|\boldsymbol{m}\| \times \|\boldsymbol{r}\|}$$

where **m** is the similarity score vector in which each entry is the BSCSE score of an example pair; **r** is the residual vector which represents the difference between the regression model and the ground truth value [12]. The default difference of positive example pairs is 1 while -1 is set for negative example pairs. Below shows the definition of the priority score.

$$S_c = \frac{\sum_{u+} \sigma(g, i\,|S, v) \cdot \vec{r}(u,*)}{\|\vec{m}\| \times \|\vec{r}\|} \cdot \beta^L$$

where *u* and *v* are the current starting and ending node; $u+$ are all positive example pairs; $\vec{r}(u,*)$ is the maximum value in the residual vector for example pairs starting from u; β is a decay factor and L is the current meta-graph layer.

After calculating the similarity scores and the priority score of the root node, FSSG extends the tree by generating all relevant paths/graphs and create a new node to represent each scenario. FSSG continues expanding the tree with the node having the highest priority score. Moreover, if there are some entries in a node where pairs are identical, it signifies that some individual paths collapse at this point. Therefore, FSSG creates a new node to represent the joint. The process of expansion repeats until a tree node reaches some destination entities. If the actual correlation value of that node is the largest, FSSG returns the meta-graph and its similarity score vector to the main programme where the BSCSE score vector will be added to a modified version of the Least-Angle Regression Model (MLAR). This expansion and addition procedure iterates until the residual vector is negligible. That symbolizes that any additional meta-graphs to be included in the MLAR will not improve the model significantly. At this stage, the discovery is completed, and at last, FSSG returns the set of meta-graphs with edge-types and their weight trained in the regression model.

In phase two, the possible node classes that match the empty spots in each graph are being recorded. Since there are many available choices for a single blank, Meng et al. [12] suggested using the node class hierarchy to resolve the issue. He proposed to take the Lowest Common Ancestor (LCA) on a class hierarchy of all possible node classes for one spot. It is because it can maximize the number of example pairs possibly being explained by each candidate. After the bottom-up transversal on the node class hierarchy and the substitution of the LCA for each missing node class, the results are finalized. FSSG incorporates this method in completing the class information in stage two. Note that ExpandGreedyTree is a function that uses a heuristic data structure **GreedyTree** and it discovers the most relevant meta-graph at one stage in the search space efficiently. Below are the pseudocodes of the algorithm. Most of the variables are defined as they are in FSPG allowing readers to reference and understand better. Interested readers may refer to [12] for additional information and explanation on the FSPG framework.

Algorithm 1 FSSG(G, S_{ep})

Input: network G, example pairs S_{ep}
Output: meta-graphs $g_{0,\ldots,k}$, weight vector $\mathbf{w}$

```
r ← {1, ..., −1}; w ← 0; k ← 0;
g_0, m_0 ← ExpandGreedyTree(G, S_ep, r);
k ← 0;
while |r| > ε do
    k ← k + 1;
    g_k, m_k ← ExpandGreedyTree(G, S_ep, r);
    X ← X ∪ m_k;
    corr ← cos(m_k, r);
    compute u, γ;
    r ← r − Xuγ;
    w ← w + uγ;
end
return g_{0,...,k}, w
```

Algorithm 2 `ExpandGreedyTree`$(G, S_{ep}, \mathbf{r})$

Input: KB G, example pairs S_{ep}, residual vector $\mathbf{r}$
Output: meta-graph g having the largest correlation with $\mathbf{r}$, and its similarity score vector $\mathbf{m}$
Data: pattern tree T

```
update priority scores of leaf nodes in T with r;
while T is expandable do
    M ← node with the largest score S_c in T;
    m ← {0, ..., 0};
    foreach tuple p in M do
        if p.(u, v) ∈ S_ep then
            m(u, v) ← σ(g', i | g, v);
        end
    end
    if m has non-zero entry then
        M.S_c ← cos(m, r);
        if M.S_c ≥ max_{T.leaf} S_c then
            g ← the meta-graph from root to M;
            break;
        end
    end
    else
        foreach tuple t in M do
            foreach pair p in t do
                foreach out-neighbor w of p.v on graph G do
                    e ← link type combination from p.v to w;
                    if M has no child N linked by e then
                        create new child tree node N of M;
                    end
                    g ← the meta-graph from root to N;
                    insert tuple ⟨(p.u, w), σ(g', i | g, w)⟩ satisfying the specific link
                    type in the link type combination to N;
                    update N.S_c;
                end
            end
        end
        foreach leaf node n in T do
            foreach entry e in n do
                if some pairs are identical then
                    if T has no this joint j then
                        create new child tree node N under the ancestors of e
                        accordingly;
                        g ← the meta-graph from root to N;
                        compute the new similarity score;
                        update N.S_c;
                    end
                    else
                        insert a new entry;
                        g ← the meta-graph from root to N;
                        compute the new similarity score;
                        update N.S_c;
                    end
                end
            end
        end
    end
end
return g, m
```

2.1.2 Demonstration

A user scenario is presented below to provide a clear illustration of the execution of FSSG.

User Scenario – Co-authorship

Step 1. Input positive example pairs by user

Step 2. Generate negative example pairs

Step 3. Initialize **GreedyTree** with a root node

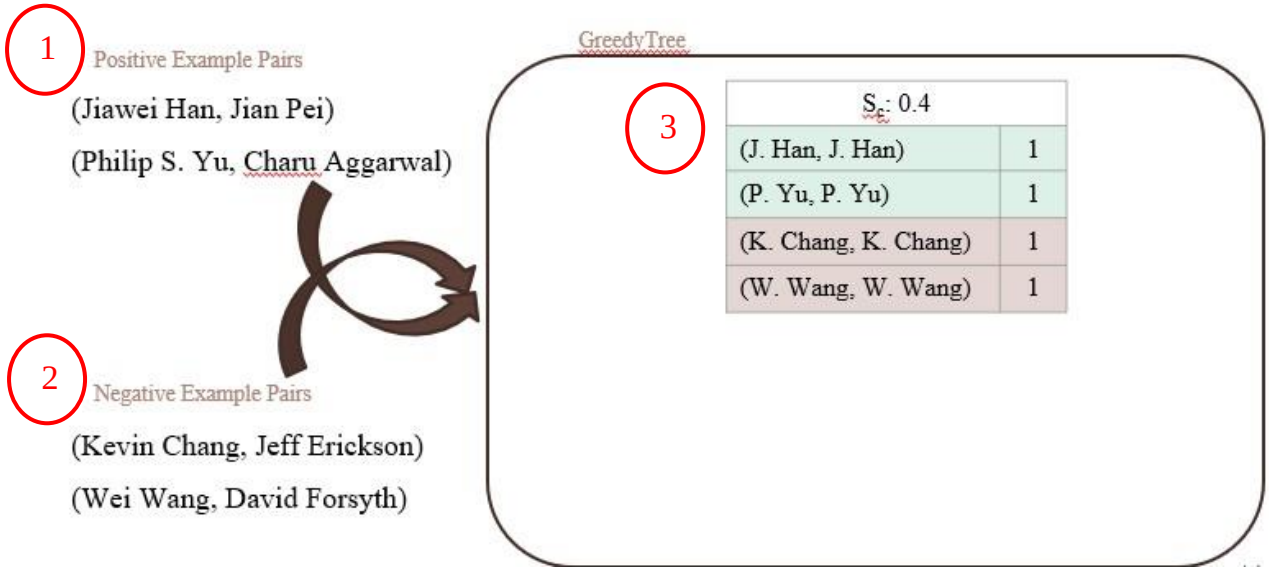


Figure 3. Step 1-3

Step 4. Expand the tree with node having the largest priority score

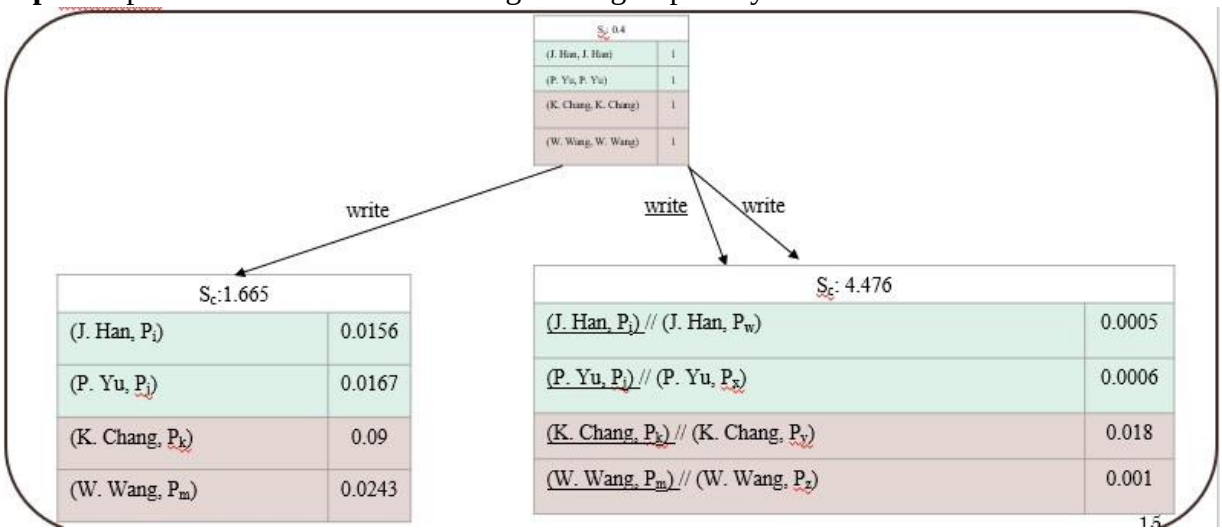


Figure 4(a) Step 4 – part I

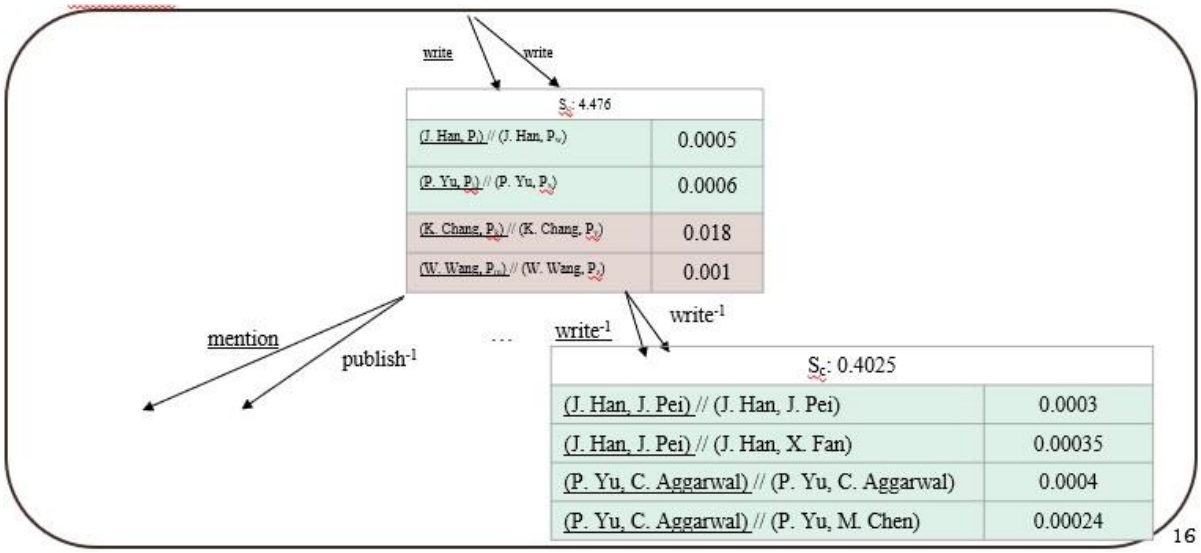


Figure 4(b) Step 4 – part II

Step 4a. If some entries containing identical pairs, create a new node

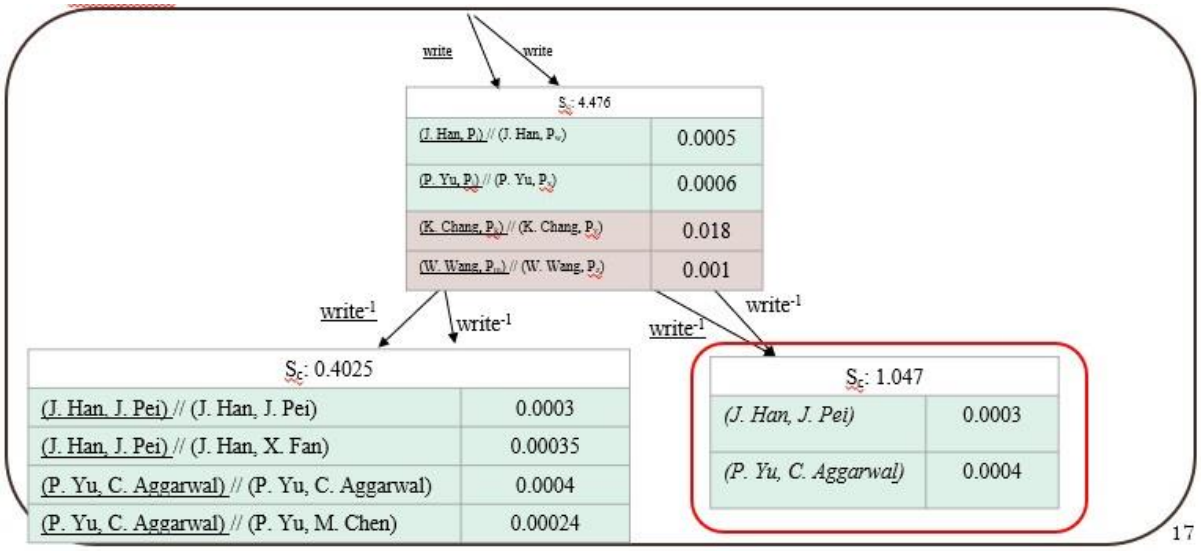


Figure 5. Step 4a

Step 5. If a node reaches some sink nodes and its priority score is the largest, returns the meta-graph with its similarity score vector

Step 6. Repeat Step 4-5 until the residual vector is negligible

Step 7. Perform LCA lookup to fill in class information

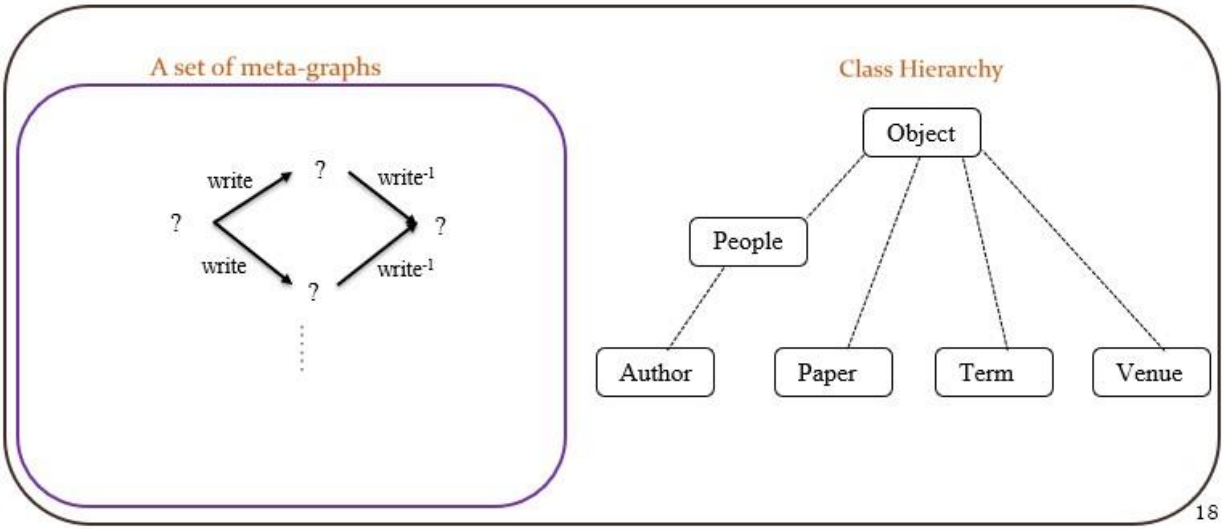


Figure 6(a) Step 7 – part I

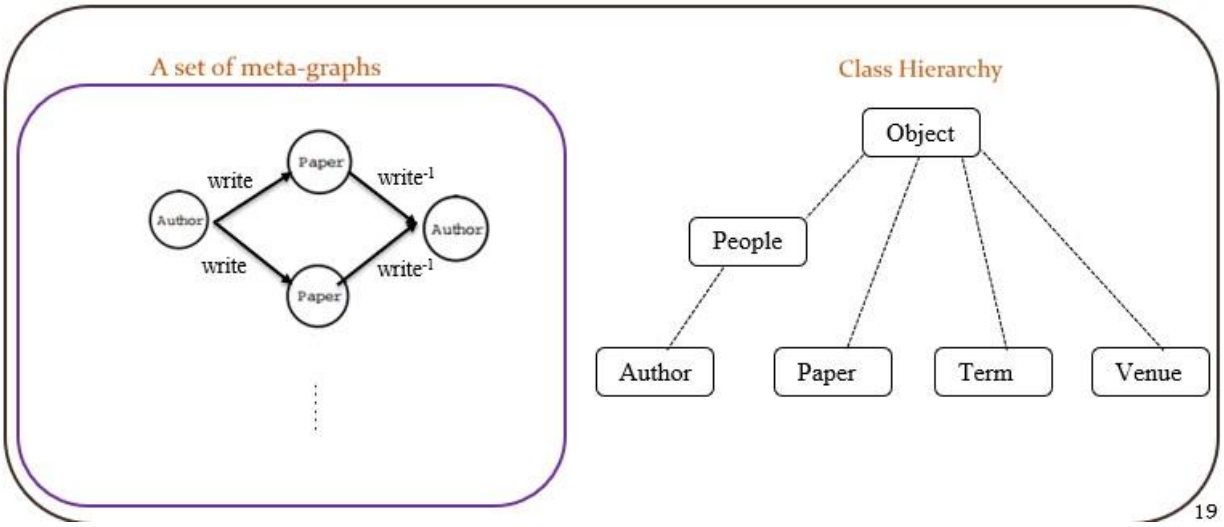


Figure 6(b) Step 7 – part II

3 Experiments and Results

The implementation of FSSG was completed, and extensive studies have been conducted to understand the performance of FSSG. This chapter focuses its discussion on some of the critical assumptions made regarding the implementation of FSSG in Chapter 3.1 and illustrates the output of FSSG mining relevant meta-graphs on the real dataset in Chapter 3.2. In Chapter 3.3, there is a detailed discussion on the setup of the empirical studies as well as a thorough analysis of the experimental results.

3.1 Assumption

As highlighted by the discussion in Chapter 2.1.1., FSSG is designed to build upon the FSPG framework allowing the algorithm to discover all types of meta-graphs. However, long and complicated meta-graphs are in fact not interesting and not meaningful nor are they easy to generate [12]. Therefore, to facilitate the effectiveness of the algorithm as well as the returned results, FSSG is being implemented in the way that it discovers only *two-branches (2B) metagraphs*. Here is the definition of 2B meta-graphs:

> Given an HIN $G = (V,E)$ and the schema $T_G = (A,R)$, a 2-braches meta-graph is a metagraph $H' = (N,M,n_s,n_t)$, where for all $x \in N$, $out\text{-}degree(x) \leq 2$.

Below diagram shows two positive examples of 2B meta-graphs and one counterexample:

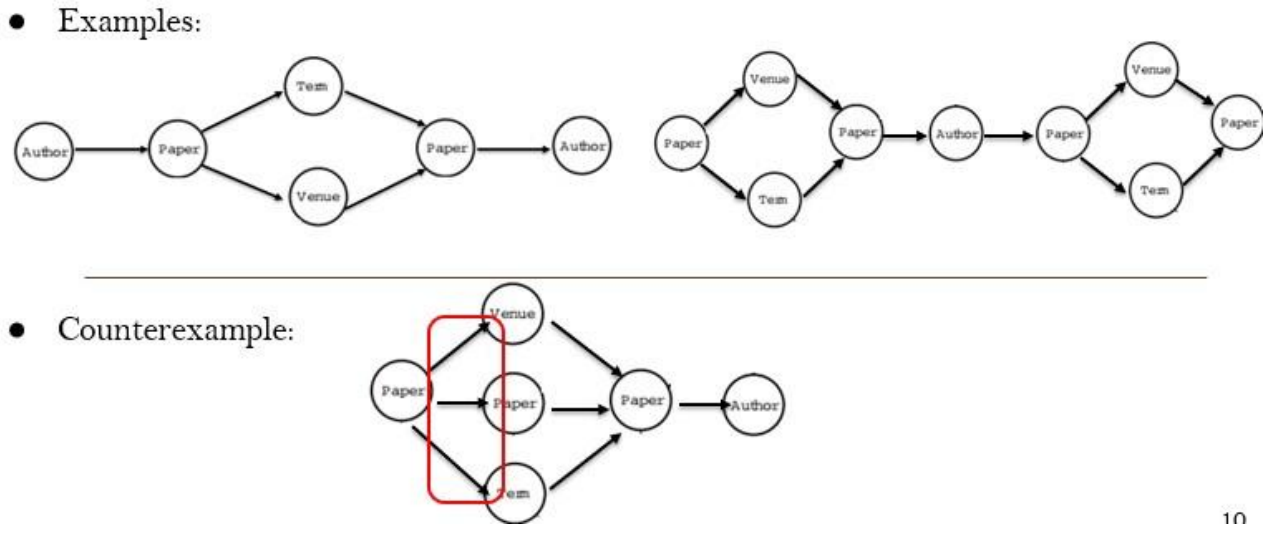


Figure 7. Examples and counterexample of 2B meta-graphs

In short, a node of a 2B meta-graph can perform either no branch or binary branch at every stage of expansion.

3.2 FSSG Implementation

To demonstrate the result FSSG have achieved, an execution on real dataset was conducted to illustrate the output. One dataset had been chosen for the execution: DBLP four area. This dataset is one of the representative datasets in the study of data mining, network mining and HIN analysis.

DBLP is a bibliography network containing computer sciences journals and conference papers. Since the entire network is enormous and most of the details are not useful for this study, a subset of this network was extracted containing sufficient data for the experimentation. The subset consists of papers published in four research areas: information retrieval, databases, data mining and artificial intelligence [12]. The schema of this subset is shown in Figure 8.

There are four edge types namely writtenBy, mentions, cites and publishedIn. Paper (P), Author (A), Venue (V) and Topic (T) are the four node classes. The subset contains more than 170000 links.

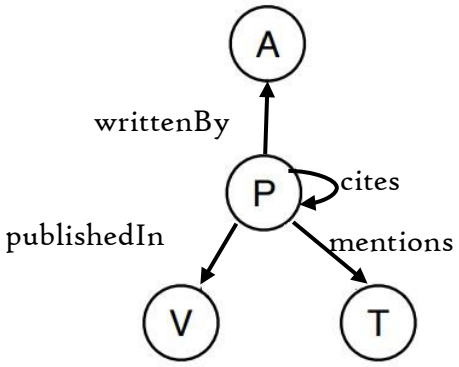


Figure 8. DBLP schema

(Rami Smair, Jamie Callan)

(Wei Wang, Philip S. Yu)

(Jiawei Han, Jian Pei)

(Wei Zhang, Rajat Mukherjee)

Figure 9. Four pairs of positive example

Four pairs of positive example were fed into the program and they are shown in Figure 9. Each pair is an instance of co-authorship, and thus the expecting result should be a set of meta-graphs directly or indirectly explaining co-authorship.

Regarding the setting of variables, as Meng suggested in [12], ε was set to 0.01 in FSSG. α was adjusted to 0.5 according to the empirical studies done in [4] and β as the decay factor was set to 0.8 [12] to avoid the search going indefinitely. The programme was written in C++ and the execution was conducted on an 4GB memory Win10 machine. The output of the execution is illustrated through Figure 10 to Figure 11.

Figure 10(a) Data Reading – Start

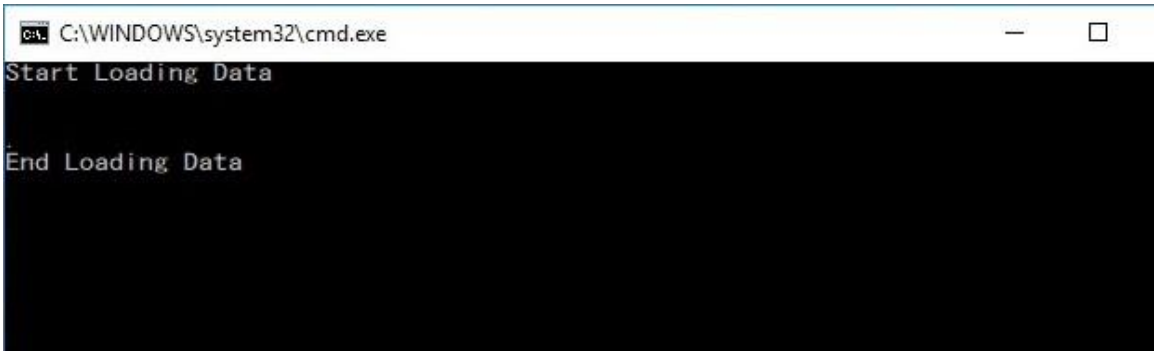


Figure 10(b) Data Reading – End

Figure 11(a) Meta-graphs generation – Part I

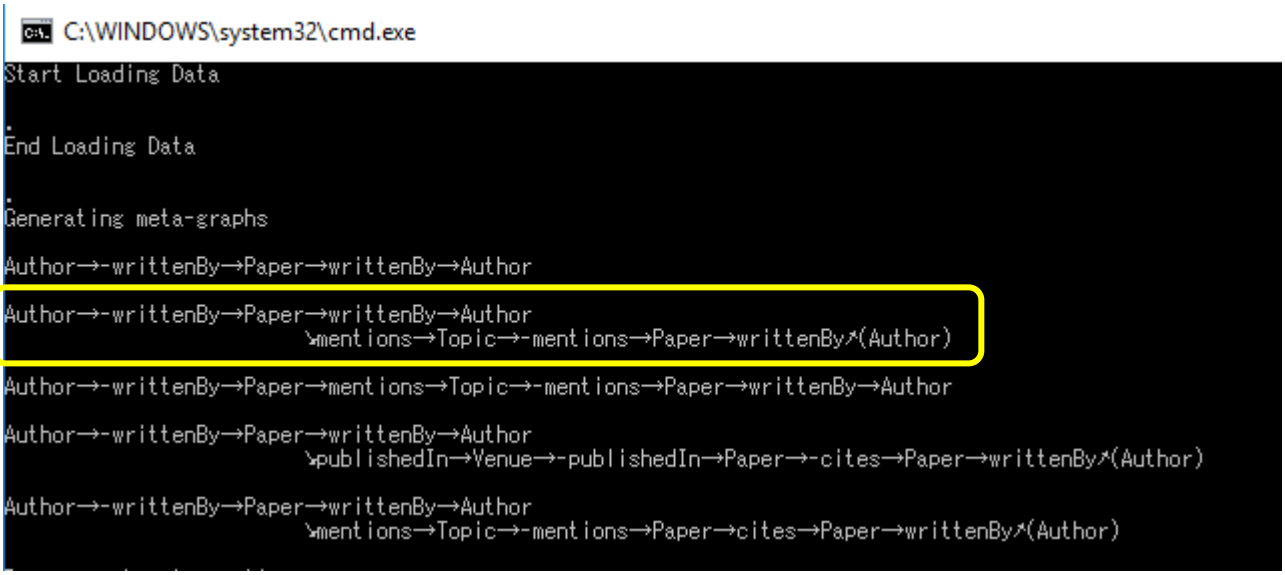
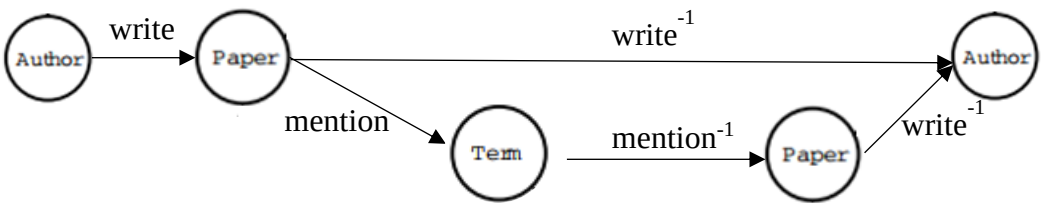


Figure 11(b) Meta-graphs generation – Part II

In this simple demonstration, the top 5 meta-graphs were returned. Among them, there are two meta-paths and three meta-graphs. The second, the fourth and the fifth structure are meta-graphs. Different kinds of arrows are used to indicate the split and the join of the branches of a meta-graph. For instance, the second structure (highlighted in Figure 11(b)) can be written as follows:



3.3 Empirical studies

Extensive experiments have been conducted to validate the effectiveness and efficiency of the proposed FSSG algorithm and its similarity model. In this chapter, the result will be discussed comprehensively. Chapter 3.3.1 is dedicated to describing the datasets being used and the setup of the experiment, including the high-level account of the procedure. Chapter 3.3.2 to 3.3.6 are subsections each devoted to discussing one aspect of performance of FSSG on the representative dataset, and these areas are effectiveness, efficiency, example pair set size, top k result and the α value in the model.

3.3.1 Datasets and setup

The empirical studies were performed on one dataset, namely the DBLP four area subset as introduced in Chapter 3.2. The team examined and analysed the effectiveness and the efficiency of the FSSG algorithm by performing link prediction task. While FSSG, as mentioned in Chapter 1 and 2, was designed to aid different data mining tasks, link prediction study provides a comparatively objective way to measure the effectiveness and efficiency of an algorithm as suggested in [12]. Moreover, performing link prediction tasks allows the team to conduct a direct comparison with previous works and to be consistent with the norm in the area.

In this work, the co-authorship relation was being studied, and the experiment was divided into to two stages, the training stage and the testing stage. The team generated a set of n positive and n negative example pairs randomly. Then, this input set was fed into the FSSG implementation. The output of this step is a group of meta-graphs generated by FSSG that directly or indirectly explains co-authorship as well as the weighting of each meta-graph. These two components contributed to the definition of the training model. Then, into the testing stage, the team randomly generated another set of example pairs with the same number of positive and negative instances (i.e. n positive, n negative). In particular, these data instances have to share the same LCA as the training data instances do. Next, the team used the trained model and the similarity function defined in Chapter 2.1.1 to compute a similarity score for each testing pair. By setting the threshold as 0.5, any positive example pair scoring 0.5 or higher and any negative example pair scoring 0.5 or lower is considered as a correct classification whereas other is regarded as a false classification. Then, ranking the pairs by their similarity score, the team can further perform various evaluations on the result to examine the performance.

Variables were set to the designated values as mentioned in Chapter 3.2. The team compared the performance of FSSG to that of FSPG to demonstrate the advantages of querying with meta-graphs.

3.3.2 Effectiveness

The team validated the effectiveness of the proposed FSSG algorithm and the similarity scoring functions by conducting link prediction task on DBLP dataset as described in Chapter 3.3.1. In this instance, 100 training pairs and 100 testing pairs were randomly generated. The result is presented in Figure 12 using a Receiver Operating Characteristics (ROC) curve. The x-axis of the ROC curve is the false positive rate whereas the y-axis is the true positive rate of the 100 testing instances.

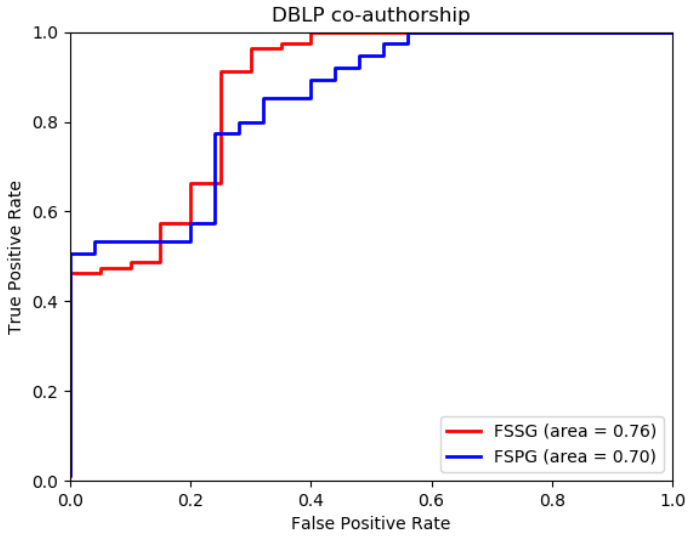


Figure 12 ROC for link prediction

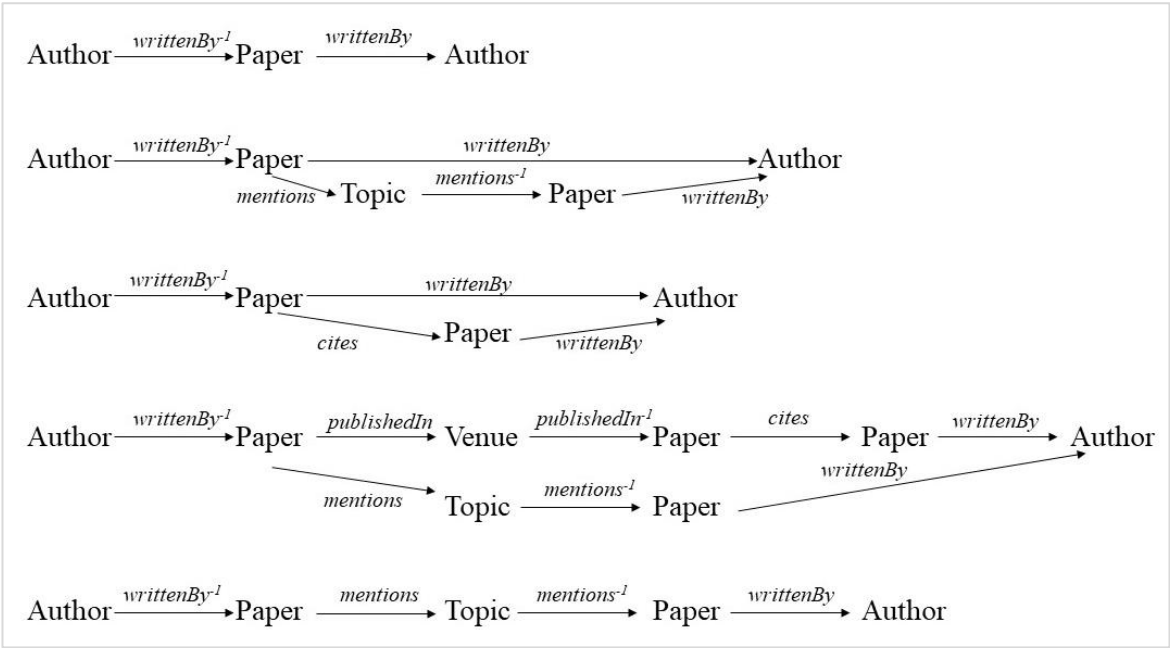


Figure 13 Top 5 relevant meta-graphs

The team compared the performance of FSSG with FSPG. Since a larger area under curve in the ROC curve indicates a higher accuracy in prediction, it is clear that FSSG discovers essential meta-graphs and is better at predicting relationship. In contrast, FSPG has a lower precision and recall as well as a smaller Area Under Curve (AUC). This observation echoed the findings in [4] where Huang highlighted the disadvantages of meta-paths. Taking the third structure in Figure 13 as an example, FSPG considers the two meta-paths separately while FSSG thinks the complete structure as one unit. The former overlooked the problem and may mistakenly classify unrelated pairs as relevant when the couple has connections for separate paths but not for the meta-graph.

Moreover, FSSG generates fewer structures than FSPG does for the training model as meta-graphs, with a more complex hierarchy, are more expressive than meta-paths. Concerning this, Figure 13 lists out the top 5 relevant meta-graphs for predicting co-authorship in DBLP. With no doubt, the meta-graph demonstrating that two authors have jointly written a paper is the top relevant predictor of the model. Additionally, FSSG discovered other interesting but not obvious meta-graphs. For instance, the second structure being shown in Figure 13 is an important and undirect co-authorship. Hence, FSSG may deploy as an expert tool helping experts identify missing meta-graphs which in terms refines the accuracy of their expert model.

3.3.3 Efficiency

In general, the time complexity of the FSSG increases in a polynomial fashion as the number of data pair increases. This is caused by the drastic surge in the number of possible meta-graphs discovered through Biased Structure Constrained Subgraph Expansion (BSCSE). Figure 14 compares the running time of FSSG and FSPG with an expanding size of input data set.

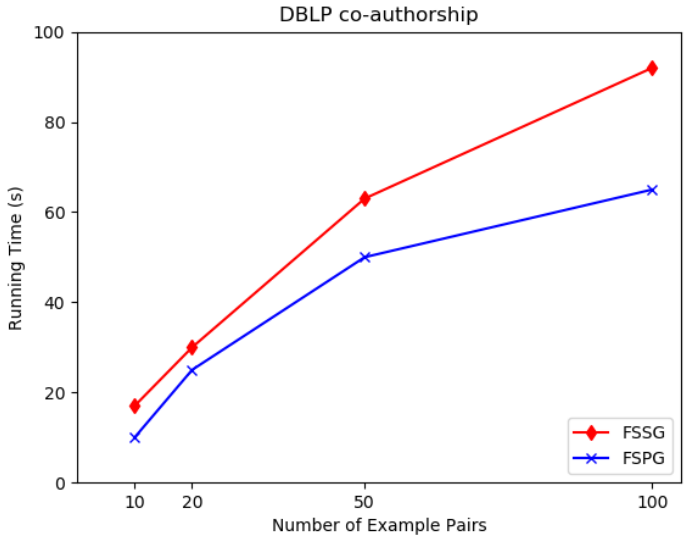


Figure 14 Running time for varying set size

The running time of FSSG is comparable to that of FSPG for small set sizes while it requires more resources for a more substantial set size. However, FSSG can discover more expressive and accurate predictors for the task as shown in Chapter 3.3.2. Thus, without sacrificing much of the time, FSSG generates more complex and truthful meta-graphs enhancing the accuracy of the model for small knowledge bases.

3.3.4 Example pair set size

The team investigated how the input data set size influences the accuracy of FSSG prediction model. It was done by contrasting the AUC values of the ROC curve of different input data set sizes for FSSG and FSPG. The result is displayed in Figure 15 below.

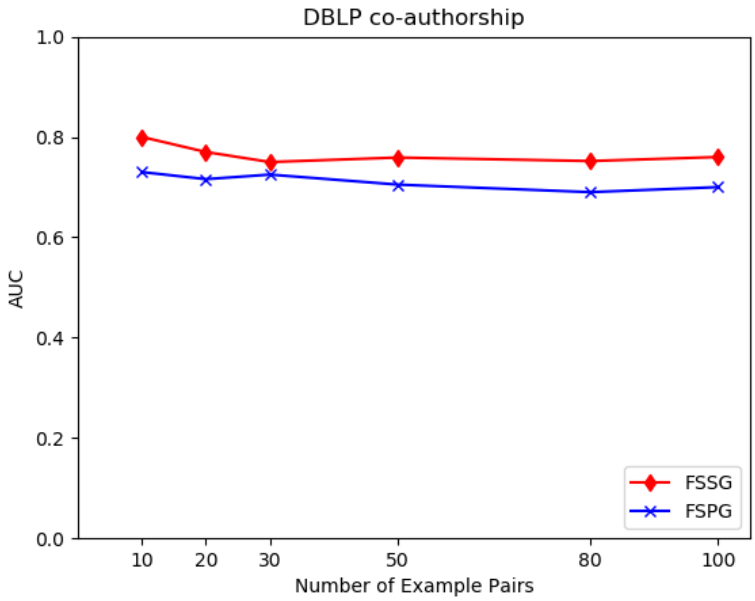


Figure 15 AUC for varying set size

With the growing number of set size, it is noticeable that the AUC values of both FSSG and FSPG are not being affected significantly. However, FSSG keeps its advantages over FSPG for various set sizes showing that FSSG is capable of discovering meaningful meta-graphs for prediction even when users provide a few data points.

3.3.5 Precision @ k

The team examined the precision of relevance classification of FSSG and FSPG at several thresholds values for input size equals 100. Figure 16 presents the performance of the two frameworks. It is worth mentioning that FSSG has a higher precision at lower k values than FSPG does. This reflects that FSSG can discover vital meta-graphs and can classify pairs into the correct group with high confidence. This also suggests that FSSG is very useful in real datasets and in aiding other realistic data mining tasks since the searching result usually will not be long for these tasks.

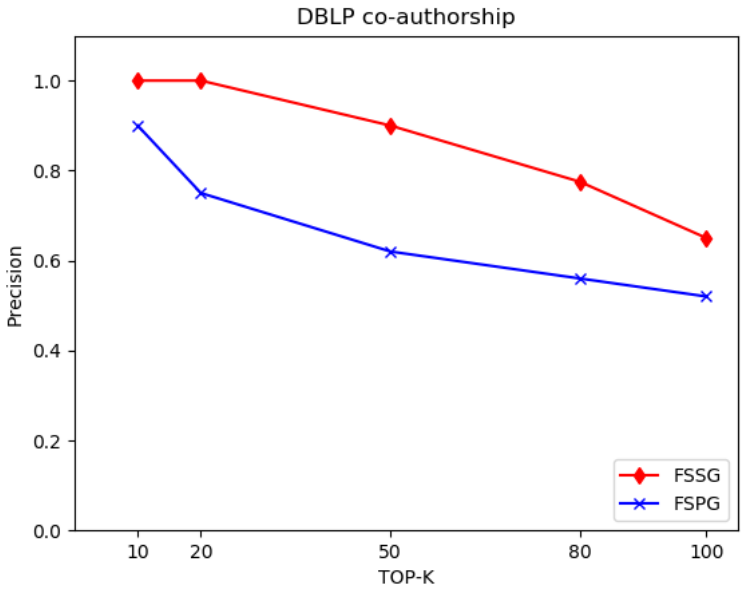


Figure 16 Precision @ k

3.3.6 α values on effectiveness

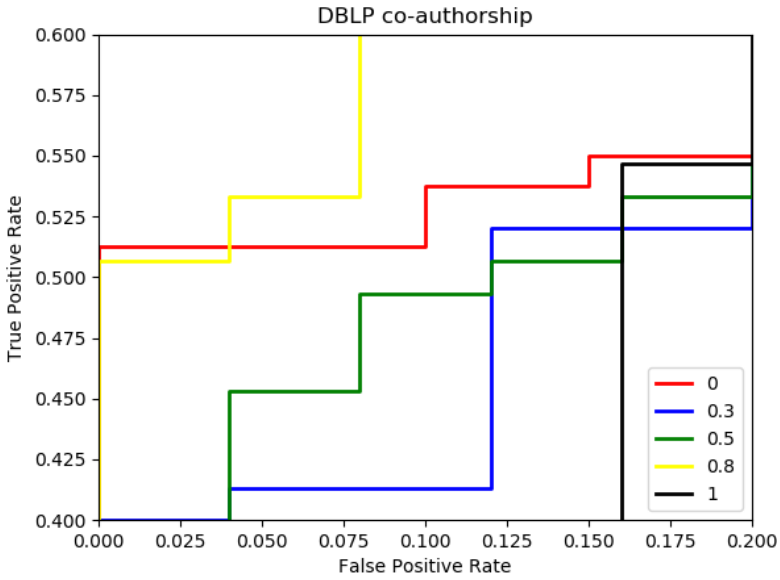


Figure 17 ROC for link prediction with different α values

There are many variables defined in the similarity functions and the framework. α, especially, plays an essential role in the definition of BSCSE [4] and the adapted version proposed in Chapter 2.1.1. The team studied the influence of α by feeding 100 data points to the algorithm. The accuracy results of different α values are illustrated in Figure 17. It is noticeable that α value has a considerable impact on the performance, and it is optimal at α=0.8. As suggested in [4] and [12], the optimal value of α varies concerning different tasks. Thus, users may find the optimal α value for their task using a for loop with small increments.

4 Difficulties encountered

During the whole project, the team was confronted by numerous theoretical and practical problems. Chapter 4.1 discusses the difficulties in understanding and replicating FSPG framework, and Chapter 4.2 shows an example of technical issue of the execution.

4.1 Impediments to implementing FSPG framework

Since FSSG employs FSPG framework as the main backbone, understanding FSPG thoroughly is required for such extension. However, the description of the FSPG framework in Meng's work [12] is limited and it became difficult to rewrite the FSPG framework from scratch. Although the team fortunately got access to the original coding of the FSPG framework, the authors of the programme did not leave any comments to explain the usage of certain enigmatic variables and the confusing flow of logic. The programme did not have any syntax or ompilation errors, it though contains severe runtime errors due to its bewildering and perplexing flow of control. To replicate and to fully extend the framework, the team spent months studying the codes line by line, rewriting the functions, removing unnecessary variables and logic. Despite the delay caused by the unexpected workload, the team has successfully rewritten the FSPG framework as part of the FSSG algorithm by late-December.

4.2 Insufficient memory capacity

Memory capacity inadequacy is a common problem of running experiments in graph mining and network mining field. It is because graphs and networks were embedded with richer information than raw text data. For example, the dataset at the minimum must store all objects, all edges, classes of each object, types of each edge and the connections. Lacking memory capacity occurs when the dataset is too large to be loaded into the main memory, the dataset is too large to be searched or the execution requires large amount of memory to support. FSSG indeed requires copious amount of memory to execute because it must store all information of each node on the **GreedyTree** as well as perform cross-checking. Some potential solutions include running the programme on a server platform providing more memory capacity or designing efficient data structures to facilitate the searching which was adopted for this project as to resolve this issue.

5 Conclusion

Meta-graph is a useful tool in many knowledge discovery tasks, e.g. similarity search, recommendation. However, currently, there has been little research done on studying effective discovery algorithms concerning meta-graphs. FSSG framework was proposed to facilitate the search of relevant meta-graphs in this study. Extensive experimental works showed that FSSG discovers not only important and expressive meta-graphs, but also other interesting meta-graphs yet being missed by domain experts. Different analyses suggested that FSSG and its similarity scoring model can generate useful meta-graphs effectively and efficiently. Some possible future works include extending the implementation to cover multi-branch expansion and deploying FSSG into real applications providing users with accurate information.

Reference

[1] X. Cao, Y. Zheng, C. Shi, J. Li and B. Wu, "Meta-path-based link prediction in schema-rich heterogeneous information network", *International Journal of Data Science and Analytics*, vol. 3, no. 4, pp. 285-296, 2017.

[2] Y. Fang, W. Lin, V. Zheng, M. Wu, K. Chang and X. Li, “Semantic proximity search on graphs with metagraph-based learning”, in *ICDE*, 2016.

[3] Z. Huang, B. Cautis, R. Cheng and Y. Zheng, “KB-Enabled Query Recommendation for Long-Tail Queries”, in *CIKM*, 2016.

[4] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis and X. Li, “Meta Structure: Computing Relevance in Large Heterogeneous Information Networks”, in *KDD*, 2016.

[5] G. Jeh and J. Widom, “Scaling Personalized Web Search”, in *WWW*, 2003.

[6] G. Jeh and J. Widom, “SimRank: A Measure of Structural-Context Similarity”, in *KDD*, 2002.

[7] H. Jiang, Y. Song, C. Wang, M. Zhang and Y. Sun, “Semi-supervised Learning over Heterogeneous Information Networks by Ensemble of Meta-graph Guided Random Walks”, *IJCAI*, 2017.

[8] X. Kong, B. Cao, P. Yu, Y. Ding and D. Wild, “Meta Path-Based Collective Classification in Heterogeneous Information Networks”, in *CIKM*, 2012.

[9] N. Lao and W. Cohen, “Relational Retrieval using a combination of path-constrained random walks”, *Machine Learning*, 2010.

[10] X. Li, Y. Wu, M. Ester, B. Kao, X. Wang and Y. Zheng, “Semi-supervised Clustering in Attributed Heterogeneous Information Networks”, in *WWW*, 2017.

[11] D. Nowell and J. Kleinberg, “The link-prediction problem for social networks”, *J. Assoc. Inf. Sci. Technol.*, 58(7), 2007.

[12] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang, “Discovering meta-paths in large heterogeneous information networks,” in *WWW*, 2015, pp. 754–764.

[13] B. Shi and T. Weninger, “Mining interesting meta-paths from complex heterogeneous information networks,” in *ICDM-MODAT*, 2014.

[14] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han, “Co-author relationship prediction in heterogeneous bibliographic networks,” in *ASONAM*, 2011, pp. 121–128.

[15] Y. Sun and J. Han, *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers, 2012.

[16] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla, “When will it happen? Relationship prediction in heterogeneous information networks,” in *WSDM*, 2012, pp. 663–672.

[17] Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu, “Pathsim: Meta path-based top-k similarity search in heterogeneous information networks,” in *VLDB*, 2011, pp. 992–1003.

[18] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han, “Recommendation in heterogeneous information networks with implicit user feedback,” in *RecSys*, 2013, pp. 347–350.

[19] H. Zhao, Q. Yao, J. Li, Y. Song and D. Lee, “Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks”, in *KDD*, 2017, pp. 634-655.

[20] Y. Zheng, C. Shi, X. Cao, X. Li and B. Wu, “Entity Set Expansion with Meta Path in Knowledge Graph”, in *PAKDD*, 201, pp. 317-329.

[21] F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledhe. In *WWW*, New York, 2007.