International Journal of Database Theory and Application
Vol.8, No.4 (2015), pp.135-148
http://dx.doi.org/10.14257/ijdta.2015.8.4.14

Mining Frequent Sub-hypergraph in an Uncertain Hypergraph for Knowledge Transfer

Xing Wang, Bin-Xing Fang, Hui He and Hong-Li Zhang

Department of Computer Science and Technology, Harbin Institute of Technology, Heilongjiang, P. R. China, 150001
yeahwx@gmail.com; {bxfang, hehui, zhl}@pact518.hit.edu.cn

Abstract

The knowledge transfer learning can generalize across domains where the types of objects and variables are different. Previous studies ignored connectivity and creativity of domain knowledge. Thus, these studies just transfer knowledge from a source domain to a target domain that not effectively use the knowledge from other domains. We proposed a method, called Multi-domain second order knowledge integration (MSKI), for integrating to address this problem. We hybridize and create new knowledge, which is formalized into an uncertain hypergraph. Then, we proposed a method to mine frequent sub-hypergraph from the uncertain hypergraph (MFS-UHG). The frequent sub-hypergraphs are pivot knowledge, which has to be transferred with high priority. We embed the pivot knowledge in the progress of MLN structure learning. The experimental evaluation on four domain datasets shows that the MSKI outperforms state-of-the-art MLN-based transfer learning.

Keywords: *Knowledge transfer, Knowledge integration, Uncertain hypergraph, Frequent sub-hypergraph mining*

1. Introduction

Different with transfer learning on features and parameters within a single domain, transfer learning on knowledge transfers the relationship between data from a source domain to a target domain, where the data are non-i.i.d (non independent and identically distributed) [1]. The transferred relationships include some concepts, such as transitivity and homophily. This process is very similar to the human learning process in which humans are even able to apply knowledge learned from one domain to another entirely different one [2]. Researches on knowledge transfer learning are mainly based on statistical relational learning, such as Markov Logic Network (MLN). Mihalkova and Mooney proposed an algorithm TAMAR and algorithm SR2LR that transfers relational knowledge with MLN across relational domains [3,4]. Davis and Domingos proposed an approach to transferring relational knowledge based on a form of second-order MLN [2].

However, previous works on knowledge transfer learning transfer knowledge in a one-to-one fashion, i.e., only from a single source domain to a single target domain. The knowledge transferred from a single source domain may not be enough to solve new problems. In contrast, Humans are far better than machines as they can learn knowledge from different domains. For example, in scientific innovation, humans get knowledge from multiple disciplines and generate new knowledge to solve new problems. What is missed in machine transfer learning is the ability to create new knowledge from different domains and to transfer pivot knowledge appearing frequently in most of domains.

In spirit of the idea, we present a new approach to transfer knowledge from multiple domains to one domain. Different with multi-task learning which learns a problem together with other related problems at the same time, using a shared representation, where the data usually are i.i.d and from same domain. Our work aimed to transfer knowledge from multiple domains to one domain, where the data are non-i.i.d and across

ISSN: 2005-4270 IJDTA

domain (e.g., between domains where the types of objects and variables are different). To achieve this intention there are two main problems we should to resolve. One is how to integrate existed domain knowledge and create new knowledge instead of just using existed knowledge within other domains. Another is how to find the pivot knowledge that appearing frequently in most of domains. These pivot knowledge should be transferred with high priority.

To tackle these problems, we proposed a method Multi-domain Second-order Knowledge Integration (MSKI), for integrating, hybridizing and creating new knowledge. The MSKI receive knowledge from multiple domains with form of uncertain hypergraph, and integrate these knowledge into a large uncertain knowledge hypergraph. Then, we propose a method to mine frequent sub-hypergraphs from the uncertain hypergraph. The frequent sub-hypergraphs can be viewed as pivot knowledge that should be transferred with high priority. Finally, we embed these pivot knowledge in the progress of MLN structure learning to transfer knowledge to the target domain. In addition, we analyzed MLN-based transfer learning mechanism and the reduction of search space by using our method. An experimental evaluation on four datasets shows that our method MSKI-MFS-UHG (MMU) is significantly outperforms the state-of-the-art single-task MLN-based transfer learning.

2. Uncertain Hypergraph

In this section, we introduced and formalized uncertain hytpergraph (intergraded knowledge). Different with previous study about mining sub-graphs in an uncertain graph database [5,6] or in a hypergraph database [7]. We present a new problem about Mining Frequent Sub-hypergraphs (pivot knowledge) in an Uncertain HyperGraph (MFS-UHG) and analyzed the complexity of this problem.

2.1. Formalization of Uncertain Hypergraph

As all graphs can be viewed as a hypergraph, the definition of uncertain graph [8] can be extended to uncertain hypergraph directly.

DEFINITION 1. An uncertain hypergraph can be represented by a 7-tuple, $\Theta_H = ((V,E),\lambda_V,\lambda_E,\mathrm{N}_V,\mathrm{N}_E,P_V,P_E)$, where (V,E) is an undirected hypergraph, V is a finite set of vertices, and E is a family of nonempty subsets of V such that $\cup_{e\in E} e = V$. N_V and N_E are the sets of vertices and hyperedges. $\lambda_V : V \to \mathrm{N}_V$ is vertex labeling function. $\lambda_E : E \to \mathrm{N}_E$ is hyperedge labeling function. $P_V : V \to [0,1]$ is a function assigning existence probability values to vertices. $P_E : E \to [0,1]$ is a function assigning existence probability values to e_i while the set $\{v_{i1}, v_{i1}, \cdots, v_{ij}\}$ in $e_i \in E$ already exists.

DEFINITION 2. An exact hypergraph $G_H = ((V',E'),\lambda'_V,\lambda'_E,\mathrm{N}'_V,\mathrm{N}'_E)$ is implicated by an uncertain hypergraph $\Theta_H = ((V,E),\lambda_V,\lambda_E,\mathrm{N}_V,\mathrm{N}_E,P_V,P_E)$, denoted by $\Theta_H \Rightarrow G_H$, if and only if $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$. The probability of an uncertain hypergraph Θ_H implicating an exact hypergraph G_H is

$$\Pr(\Theta_H \Rightarrow G_H) = \prod_{v \in V'} \Pr_V(v) \cdot \prod_{v \in V \setminus V'} (1 - \Pr_V(v')) \cdot \prod_{e \in E'} \Pr_E(e) \cdot \prod_{e \in E \cap (V' \times V') \setminus E'} (1 - \Pr_E(e'))$$

Theorem 1. For any uncertain hypergraph $\Theta_H, (Imp(\Theta_H), 2^{Imp(\Theta_H)}, P_{\Theta_H})$ is a probability space. (proof omitted)

3. Frequent Sub-Hypergraph Mining

The different between our problem and [8] are as follows: (1) this paper is aimed at

$$\Pr(\Theta_H \Rightarrow G_H) = \prod_{v \in V'} \Pr_V(v) \cdot \prod_{v \in V \setminus V'} (1 - \Pr_V(v')) \cdot \prod_{e \in E'} \Pr_E(e) \cdot \prod_{e \in E \cap (V' \times V') \setminus E'} (1 - \Pr_E(e'))$$

formalizing uncertain hypergraph rather than uncertain ordinary graph. (2) this study is focused on mining frequent sub-hypergraph from a single uncertain large hypergraph instead of a graph database.

DEFINITION 3. An exact hypergraph $G_H = ((V,E), \lambda_V, \lambda_E, \mathrm{N}_V, \mathrm{N}_E)$ is sub-hypergraph isomorphic to another exact hypergraph $G'_H = ((V',E'), \lambda'_V, \lambda'_E, \mathrm{N}'_V, \mathrm{N}'_E)$ denoted by $G_H \sqsubset G'_H$, if there exists an injection $\phi: V \to V'$ such that (1) $\lambda_V^{G_H}(v) = \lambda_{V'}^{G'_H}(\phi(v))$ for every $v \in V$, (2) $\{\phi(v) : v \in e\} \in E'$ for every $e \subseteq V$ if $e \in E$, (3) $\lambda(\{v : v \in e\}) = \lambda'(\{\phi(v)\})$ for every $e \in E$.

DEFINITION 4. A connected exact hypergraph G_H^C is asub-hypergraph in a large uncertain hypergraph Θ_H^L if G_H^C is sub-hypergraph isomorphic to at least one implicated hypergraph in Θ_H^L.

We apply Minimum Number of vertex Images (MNI) support [9] to hypergraph.

DEFINITION 5. Let G_H and $G_{H'}$ be two certain hypergraphs and let F be the set of all sub-hypergraph isomorphism of G_H to $G_{H'}$. The MNI support of G_H w.r.t $G_{H'}$ is defined as $\min_{e \in E} |\{e' \in E' \mid \exists f \in F : f(e) = e'\}|$.

DEFINITION 6. Given a large uncertain hypergraph Θ_H^L, let $Imp(\Theta_H^L)$ be a set of all certain hypergraphs implicated by Θ_H^L. The support of a sub-hypergraph in

$$\begin{bmatrix} s_1 & s_2 & \cdots & s_m \\ P(s_1) & P(s_2) & \cdots & P(s_m) \end{bmatrix}$$

Θ_H^L is a probability distribution:

, where $s_1, s_2, \ldots, s_m$ are the supports defined by definition 5.

DEFINITION 7. Let $g = \{G_H^L \in \mathrm{Imp}\}$, the MNI expected support of a sub-hypergraph S in Θ_H^L is

$$esup(S) = \sum_g sup_{\Theta_H^L}(S) \cdot P(\Theta_H^L \Rightarrow G_H^L) = \sum_g |MNI| \cdot P(\Theta_H^L \Rightarrow G_H^L).$$

If *esup*(S) is greater than threshold, then S is a *minsup* expected frequent sub-hypergraph.

Lemma 1. It is an NPC problem to mine all expected frequent sub-hypergraphs in an uncertain large graph. (Detailed proof can be found in the appendix)

Theorem 2. It is an NPC problem to mine the patterns of all frequent sub-hypergraphs in an uncertain large hypergraph for an arbitrary expected support threshold. (Proof see appendix)

4. Knowledge Integration

In this section, we propose a method, called Multi-domain Second-order Knowledge Integration (MSKI), for integrating, hybridizing and creating new knowledge, which is generated an uncertain hypergraph from different domains.

In order to integrate knowledge effectively and realistically, we select the Second-Order Logic Template (SOLT) [2] as the representation of knowledge. Consistent with the idea of MLN, we consider that a SOLT is not always true. We can associate it with a probability for expressing domain generalization possibility. Our knowledge integration method has five stages: selection, mapping, conversion, connection, and mutation.

Selection: This step chooses some high frequency First Order Logic (FOL) to simplify the complexity of the integration. We use motifs as FOLs created and selected by algorithm LSM.

Mapping: It is impossible to integrate motifs from different domains, unless each pair of motifs has at least one same predicate and constant symbols.

So we use the algorithm proposed by [3] to get a predicate mapping. We mark hyperedges and vertices in our motifs with the same label according to mapping. For example, one of the mappings is $R: actor \longrightarrow adviseBy \longrightarrow workUnder$ which means the three different predicates from different domains can be mapped to R.

Conversion: Directly integrating these motifs will lead to a complex uncertain hypergraph and break the probability integrity. The probability integrity in our semantic environment means that the basic unit of the probability is a formula rather than a predicate.

Therefore, we treat motifs as an entirety with a probability (the ratio of the number of true grounds to the number of all possible grounds), and convert the FOLs to SOLTs by replacing predicate names with variables.

Connection: We embed new knowledge into existing knowledge by overlap the largest common part of them.

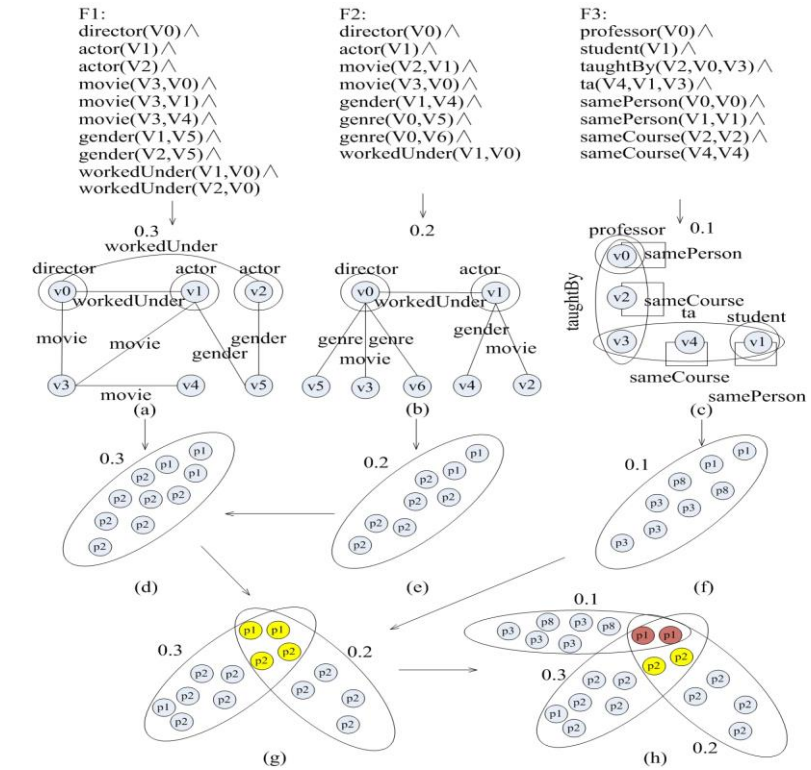
Mutation: It will generate more and more hyperedges through the connection process while do not create any new structure of the knowledge. Thus, we have to mutate the remaining of knowledge and eliminate the hyperedges with low probability.

An example of integration process is shown in Figure 1. Suppose we selected three FOLs F_1, F_2, F_3. Each FOL can be viewed as a hypergraph shown in Figure 1(a), (b), (c). The hypergraph Fig.2(a) can be converted into a hyperedge Figure 1(d). Both of them have the same probability and are formed by couples of vertices transferred from predicate hyperedges in Figure 1(a). The label of the hyperedge Fig.1(d) is the Depth First Search(DFS) cannoail code.

According to the conversion above, we can transfer the motifs from multi-domains to the relevant SOLT hyperedges. As shown in Fig.1(d) and Figure 1(e), they represent two SOLT hyperedges from the same domain IMDB. In order to integrate Figure 1(e) into Figure 1(e), we found the maximum common vertexes $P1, P1, P2, P2$ by gradually comparing their sub-hypergraphs' DFS cannoail codes that represent their relation of constants variable in motif and embed them into a new hypergraph Figure 1(g). Then we added the hyperedge Figure 1(f) converted from domain UW-CSE to hypergraph Fig.1.left(g) to generate a hypergraph Figure 1(h).

We adopt three mutation operations to create new knowledge. (1) Non-common part of hyperedge is deleted with a given probability P_d. (2) Non-common part of hyperedge priority is embedded in the common part of hypergraph with a given probability P_c. (3) If some vertices have the same SOL predicate type in the non-common part of hypergraph and are dissatisfied with operation (2), they can be randomly embedded in the non-common part of hypergraph. An example of knowledge mutation is shown in Figure 2 It will generate two possible mutated uncertain hypergraph Figure 2(d) or Figure 2(e) by adding SOLT hyperedge Figure 2(b) to uncertain hypergraph Figure 2(a).

Figure 1. An Example of Knowledge Integration of Three Motifs from Two Domains



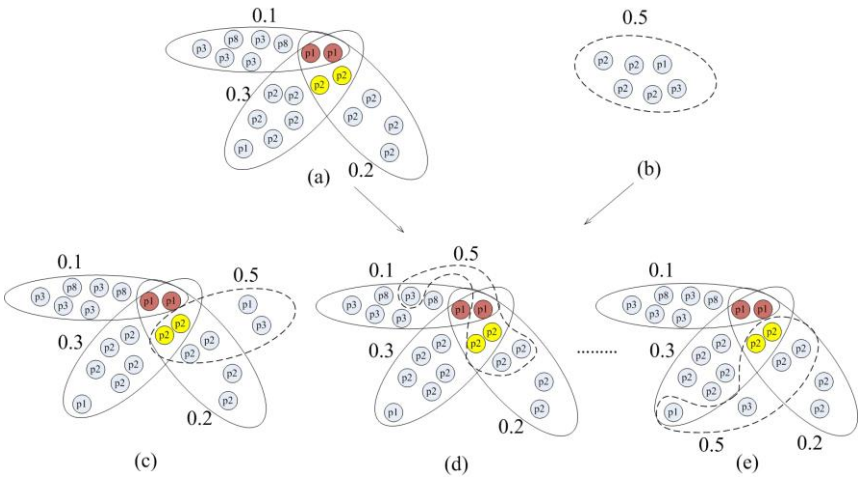


Figure 2. An Example of Knowledge Mutation

5. Mining Frequent Sub-Hypergraph

In this section, we propose a hypergraph DFS canonical coding scheme to avoid redundant search for candidate sub-hypergraphs. Then, we design an exact algorithm for computing MNI support for small instances and an approximation algorithm for efficiently computing the MNI support.

5.1. Hypergraph DFS Cannoail Code

One of the most promising ways to avoid a redundant search is to define a canonical description of a graph. However, a canonical form of ordinary graph cannot be applied to hypergraphs directly, because the way of depth-first search in hypergraphs is different from that in ordinary graphs. Therefore, we need to extend the ordinary graph canonical algorithm onto hypergraph.

5.1.1. Hyperedge Set: After performing a depth-first search in a hypergraph, we construct a DFS hypergraph. It is clear that one hypergraph can have several different DFS hypergraphs with different starting points and growing edges.

For example, hypergraphs in Fig.3(b)-(d) are isomorphic to that in Fig.3(a). We use subscripts to label this order according to their discovery time. If $i < j$, then v_i is discovered before v_j. The thickened hyperedges in Fig.3(b)-(d) represent three different DFS hypergraphs for the hypergraph in Fig.3(a).

Due to different ways of traversing the hypergraph, hyperedge set cannot be simply divided into forward ones and backward ones as ordinary graph. We classfiy hyperedge set into four types: a single hyperedge having only one vertex; a F&B hyperedge linking both the traversed vertices and the new vertices; a forward hyperedge only connecting the new vertexes; a backward hyperedge only connecting the traversed vertices.

For simplicity, $\{v_0,\cdots,v_i,\cdots,v_j,\cdots,v_n\} \in \varepsilon$ is an ordered set representing a hyperedge. We define $HE_{f,I} = \{e \mid \forall i, j, i < j, e = \{v_0,\cdots,v_i,\cdots,v_j,\cdots,v_n\}\}$ as the forward edge set in G_H^I, and $HE_{b,I} = \{e \mid \forall i, j, i > j, e = \{v_0,\cdots,v_i,\cdots,v_j,\cdots,v_n\}\}$ as the backward edge set in G_H^I. The thickened black hyperedges in Fig.3(b)-2(d) represent the forward ones; the blue hyperedges represent the F&B; the dash hyperedges represent the backward; the red color represents the single. If any adjacent elements in the set satisfies $v_i < v_j$, it is forward, otherwise it is backward. If exist i and j satisfying $v_k < v_i$ for every $k < i$, $v_k > v_j$ for every $k > j$, and $i < j$, then it is F&B. If the set has only one vertex, it is single.

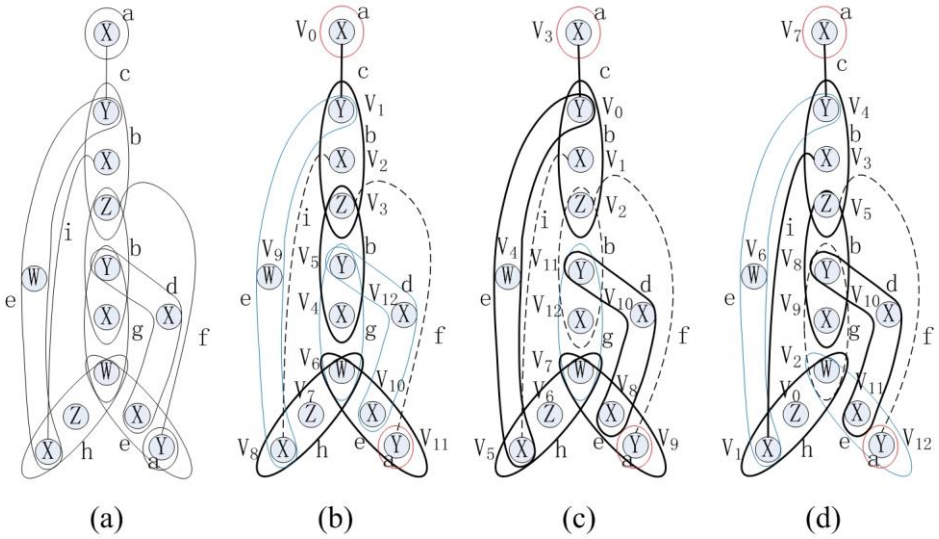


Figure 3. Depth-First Search Hypergraph and its Forward/Backward/F&B/Single Hyperedge Set

Table 1. Hypergraph DFS Code for Fig.3(b), (c), (d)

H.N	(b)	(c)	(d)
0	({0},{X},a)	({0,1,2},{Y,X,Z},b)	({0,1,2},{Z,X,W},h)
1	({0,1},{Y,X},c)	({0,3},{Y,X},c)	({1,3},{X,X},i)
2	({1,2,3},{Y,X,Z},b)	({3},{X},a)	({3,4,5},{X,Y,Z},b)
3	({3,4,5},{Z,X,Y},b)	({0,4,5},{Y,W,X},e)	({4,6,1},{Y,W,X},e)
4	({4,6,5},{X, W,Y},g)	({5,1},{X,X},i)	({4,7},{Y,X},c)
5	({6,7,8},{W,Z,X},h)	({5,6,7},{X,Z,W},h)	({7},{X},a)
6	({8,2},{X,X},i)	({7,8,9},{X,X,Y},h)	({5,8,9},{Z,Y,X},b)
7	({8,9,1},{X,W,Y},e)	({9},{Y},a)	({9,8,2},{X,Y,W},g)
8	({6,10,11},{W,X,Y},e)	({9,2},{Y,Z},f)	({8,10,11},{Y,X,X},d)
9	({11},{Y},a)	({8,10,11},{X,X,Y},d)	{11,12,2},{X,Y,W},e)
10	({11,3},{Y,Z},f)	({11,12,7},{Y,X,W},g)	({12},{Y},a)
11	({10,12,5},{X,X,Y},d)	({12,11,2},{X,Y,Z},b)	({12,5},{Y,Z},f)

We define four partial orders, $\prec_{f,I}$ on $HE_{f,I}$, $\prec_{b,I}$ on $HE_{b,I}$, $\prec_{s,I}$ on $HE_{s,I}$ and $\prec_{fb,I}$ on $HE_{fb,I}$. Assume $e_1 = \{v_{0_1},\cdots,v_{i_1},\cdots,v_{j_1},\cdots,v_{m_1}\}$, $e_2 = \{v_{0_2},\cdots,v_{i_2},\cdots,v_{j_2},\cdots,v_{n_2}\}$. Then, (1) $e_1 \prec_{s,I} e_2, \forall e_1, e_2 \in HE_{s,I}$, if and only if $0_1 < 0_2$. (2) $e_1 \prec_{f,I} e_2, \forall e_1, e_2 \in HE_{f,I}$, if and only if $\max(i_1) < \max(i_2)$. (3) $e_1 \prec_{b,I} e_2, \forall e_1, e_2 \in HE_{b,I}$, if and only if either of the following holds: $(i) \exists t, 0 \le t \le \min(m,n), k_1 = k_2 \ for\ k \le t, \quad . \quad t_1 < t_2$ $(ii)\ k_1 = k_2 \ for\ 0 \le k \le m, and\ n \ge m$. (4) $e_1 \prec_{fb,I} e_2, \forall e_1, e_2 \in HE_{fb,I}$ if and only if $\max(i_1) < \max(i_2)$.

Theorem 3. The relation $\prec_{HE,I}$ defined by combining the partial orders (1-4) is a linear order on *HE*. (proof omitted)

5.2. Hypergraph DFS Code and Lexico Order

5.1.1. Hyperedge Set:

Definition 8. Given a DFS numbering for a hypergraph G_H, a hyperedge sequence $(G_H, I) = (e_i)$ can be constructed based on $e_i \prec_{HE,I} e_{i+1}$, where $i = 0,\ldots,|HE|-1$. (G_H, I) is defined as a hypergraph DFS code.

The Hypergraph DFS code for Fig.4 (b), (c), and (d) as shown in table 1. In order to construct a DFS lexicographic order of hypergraph, we denoted a hyperedge with label $l(v_0,\cdots,v_n)$ as $\{v_0,\cdots,v_n\}$, vertices as labels $\{l(v_0),\cdots,l(v_n)\}$; combined all of them into a 3-tuple: $(\{v_0,\cdots,v_n\},\{l(v_0),\cdots,l(v_n)\},l(v_0,\cdots,v_n))$.

Definition 9 (DFS LexicoHyperGraphic order) Define two linear orders $\prec_{VL}$, $\prec_{EL}$ in the vertex label set (*VL*) and hyperedge label set (*EL*) respectively. The LexicoHyperGraphic combination of $\prec_{VL}$, $\prec_{EL}$ and $\prec_{HE,I}$ is a linear order $\prec_{he}$ on the set $HE_I \times VL \times EL$. DFS LexicoHyperGraphic order is a linear order defined as follows. If $\alpha = code(G_H^{\alpha}, I^{\alpha}) = (\alpha_0, \alpha_1, \ldots, \alpha_m)$ and $\beta = code(G_H^{\beta}, I^{\beta}) = (\beta_0, \beta_1, \ldots, \beta_n), \alpha, \beta \in Z$ then $\alpha \le \beta$ if and only if either of the following is true. (1) $\exists i, 0 \le i \le \min(m,n), \alpha_j = \beta_j$ *for* $j \le i, \alpha_i \prec_{he} \beta_i$ (2) $\alpha_j = \beta_j$ *for* $0 \le k \le m$ *and* $n \ge m$.

Definition 10(Minimum DFS code) Given a hypergraph G_H, $Z(G_H) = \{code(G_H, I) | \forall I, I \text{ is DFS tree for } G_H\}$, based on DFSLexicoHyperGraphic order, the minimum one, $\min(Z(G_H))$, is called Minimum DFS code of G_H. It is also the canonical label of G_H.

Theorem 4. Given two hypergraphs G_H and $G_{H'}$, G_H is isomorphic to $G_{H'}$ if and only if $\min(G_H) = \min(G_{H'})$. (proof omitted)

5.2. Approximation Algorithm for Computing Expected Support

Given a sub-hypergraph S, an uncertain hypergraph Θ_H^L, a certain hypergraph $G_H^{L'}$ obtained by removing uncertainly all vertices and hyperedges from Θ_H^L. Let the set $ED = \{S_1, S_2, \ldots\ldots, S_m\}$ be all embeddings of S in $G_H^{L'}$. According to definition 8, we must compute over all $2^{|E_{\Theta_H}|}$ implicated graphs of Θ_H^L to get the result of $esup_{MNI}(S)$.

Due to the exponential time complexity. We proposed an approximation algorithm to efficiently compute $esup_{MNI}(S)$.

The idea of the algorithm is similar with literature [6], but the difference is that it uses the fully polynomial randomized approximation scheme (FPRAS). The FPRAS is based on binary variable and Poisson experiment. However, according to the definition of $esup_{MNI}(S)$, the variable is not binary but ranges in $[0,1,2,\ldots,\max(MNI)]$.

Algorithm 1: APPROX-MNI-SUP

Input: $S, \Theta_H^L, \{S_1, S_2, \ldots, S_t\}, \varepsilon, \delta$

Output: *approximate* $esup_{MNI}(S)$

1 $esup_{MNI}(S) \leftarrow 0$

2 *Construct Disjunctive Normal Form*(*DNF*) *formula* $F = C_1 \vee C_2 \vee \cdots \vee C_n$

3 $N \leftarrow r^2t^2\ln(2/\delta)/2\varepsilon^2$
4 $Z \leftarrow \Pr(C_1)+\Pr(C_1)+\cdots+\Pr(C_n)$
5 $U \leftarrow \sum_{i=1}^{t}|SC_i|$
6 $X \leftarrow 0, Y \leftarrow 0$
7 **for** $j=1$ **to** N **do**
8 i *is a random integer follow by probability* $|SC_i|/U$
9 λ *is a random truth assignment satisfying* C_i
10 **if** λ *doesn't satisfy* C_j *for all* $1 \le j \le i-1$ **then**
11 $m \leftarrow MNI(\lambda)*\Pr(\lambda)$
12 $Y \leftarrow Y+m$
13 end
14 return

Theorem 5. Suppose $X_1,\ldots,X_m$ is i.i.d random variables, satisfying $X_i \in \{1,2,\ldots,r\}$, $\Pr(X_i=k)=P_k$. Let $X=\sum_{i=1}^{m}X_i, \mu=\mathrm{E}[X]=\sum_{i=1}^{m}E[X_i]$. If $n=\lceil r^2t^2\ln(2/\delta)/2\varepsilon^2\rceil$, then $\Pr(|1/m\sum_{i=1}^{m}X_i-\mu|\ge\varepsilon\mu)\le\delta$, that is n samples provided a (ε,δ) approximation.

The analyzed of time complexity of APPROX-MNI-SUP as follow. Lines 7 to 13 is loop for $N=r^2t^2\ln(2/\delta)/2\varepsilon^2$ times, we have to compute $MNI(\lambda)$ for $O(k^2c^2)$ times in line 11, where c is the number of hyperedges in S. The expected time complexity of APPROX-MNI-SUP is $O(k^2c^2N)$.

The complete algorithm MFS-UHG is similar with gSpan[10]. The distinctions are our support computing, the way of hypergraph DFS code growing and code pruning. Due to space limitation, we omit the describtion of MFS-UHG. Sub-hypergraph isomorphism is an NP-complete problem. Therefore, the runtime of MFS-UHG should be exponential. If measured by the number of sub-hypergraph and hypergraph isomorphism tests, the runtime can be bounded by $O(kF+rF)$, where k is the maximum number of duplicate codes of a frequent sub-hypergraph that grow from other minimum codes. F is number of frequent sub-hypergraphs.

6. Knowledge Transfer

The pivot knowledge mined and refined from source domains needs to be transferred for improving the target domain learning. A method of transfer knowledge is declarative bias. Davis and Domingos proposed to make use of declarative bias of Inductive Logic Programming (ILP) system to restrict the search space, further, refining the clauses picked by the greedy procedure to better match the target domain [2]. Similar to this system, there are two steps to transfer knowledge to the target domain.

Second Order Template (SOT) generalization: We pick the refined pivot knowledge that has at least one true ground in the target domain, and replace all labels of pivot with the target predicate.

Declarative bias: We try all combinations of sign flips of literals in a clause and greedily form a best MLN, which gives the highest WPLL. The MLN serves as the seed network during Markov logic structure learning (MSL) [11]. The beam search can greedily produce a candidate clause from the seed network to match the target domain better. This process can find more clauses in accordance with the target domain.

Theorem 6. Suppose the target domain has N predicates, the j^{th} extension of a formula that composed by i predicates has D_i^j type consistent mapping (if we do not consider the trim of equivalent modulo variable renaming during the search space generating). Suppose a SOT F_a has n predicate variables. It FOL generation of the target domain is a set $F_{G(a)}$. The i^{th} of $F_{G(a)}$ has m_i FOL predicates. Let A be a set of SOT mining from source domains. If using strict declarative bias (only use the FOL generation from source domains as structure MLN of target domain) to transfer A to a target domain \$G\$, then the formula search space reduction of target

$$d = N + \sum_{i=2}^{N} \sum_{j=1}^{C_N^i + N} 2^{D_i^j} \cdot 2^i - \sum_{i=1}^{|A|} 2^{n_i} \cdot \prod_{j=1}^{n_i} m_i^j$$

domain is

Although the above analyses are under strict restrictions, this result reflects that these method of transfer learning can reduce the clause search space greatly.

7. Experiment

In this section, we carried out experiments to investigate whether our algorithm is better than other approaches. This experiments use four different datasets to evaluate the algorithms that are described in this paper. These datasets are publicly available at http://alchemy.cs. washington.edu. The details are shown in Table 2.

Table 1. Data Set

Data Set	Consts	Types	Preds	True Gliterals	Total Gliterals
IMDB	316	4	10	1540	32615
UW-CSE	1323	9	15	2673	678899
WebKB	1700	3	6	2065	688193
Yeast	3079	5	10	42558	687422

IMDB dataset. Mihalkova and Mooney created from the IMDB.com database, describes a movie domain [3]. It contains relationships among movies, actors and directors. For instance, WorkedIn(person,movie), Actor(person), etc. The data is split into five disjoint folds. **UW-CSE dataset**. Richardson and Domingos describes the Department of computer Science and Engineering at the University of Washington [12]. Its predicates describe students, faculty, and their relationships.the data is split into five folds. **WebKB dataset**. The dataset contains web pages from four universities labeled according to the entity they describe.The data from each university is treated as a separate fold. **Yeast dataset**. This dataset contains information on protein interactions and protein complex data. We used the version of the data from literature [2], which is split into four disjoint subsamples which are used as folds.

In the first baseline, we applied TARMAR to multi-task transfer learning fasion directly. However, the TARMAR is designed to transfer knowledge from single domain to a target domain. Therefore, we combined the databases from different domains by hand, then transferred the integrated knowledge to target domain. The second baseline is the DTM algorithm. We chose ten best SOLTs from each domain transferred mutually. In order to increase the accuracy of transfer, the integrated forty SOLTs would be transferred into a target domain. The third baseline is the state-of-the-art MLN structure learning algorithm LSM [13].

Each dataset was divided into four independent folds, on which we performed leave-one-out cross-validation, training on every subset of three folds and testing on the fourth. The results represent averages over the four folds from each domain. Note that the

original IMDB and UW-CSE dataset have five fold, as [14] we used only the first four folds in order to maintain consistency with our use of WebKB and Yeast. MSL structure refinement was time-limited to 48 hours for each trial. The LSM parameter values follow the set of [13]. The MSKI-MFS-UHG (MMU) parameter values were $P_d = 0.1, P_c = 0.5.$ The probability (frequency of domain generalization possibility) on a hyperedge (SOT) is usually small. Thus, we set the support threshold with the form of percentile 0.2. For both datasets, we performed inference over the groundings of each predicate to compute their probabilities of being true, using the groundings of all other predicates as evidence. This accounts for the differences in our results from those reported by [15] and [13]. For evaluating the result of the contrast experiment, we use the two metrics employed by literature [11], the area under the precision-recall curve (AUC) and the conditional log-likelihood (CLL).

Table 2. Some Frequent SOTs

r(x,y)r(y,x) s(y,z) s(z,x) r(z,y) t(x,z)
r(x,y)r(x,z)s(x,x)t(x,y)p(x)
r(x,y) r(z,y) s(x,z) s(x,x)t(x)
r(x,y)r(z,y)s(x,x)s(x,z)

In practically, we get average 27 hyperedges after MSKI from three domains in turn. Thus, the time consuming of our MMU is not too much. The following formulas are some pivot structure of knowledge mining from the integrated knowledge hypergraph. These SOT are break the length limitation of DTM, which have only two or three literals, and expanded seed network of DTM and candidate motifs of LSM.

Table 3. Experimental Results Comparing MMU to MSL, LSM, TRMAR and DTM

Algorithm	**AUC**				**CLL**			
	I	U	Y	W	I	U	Y	W
MMU(LSM)	**0.71**	**0.21**	**0.19**	0.51	**-0.05**	**-0.03**	**-0.33**	-0.10
MMU(MSL)	0.51	0.20	0.17	0.49	-0.11	**-0.03**	-0.34	-0.13
DTM(C)	0.46	0.19	0.15	0.41	-0.15	-0.04	-0.39	-0.30
DTM(I)	-	0.15	0.13	0.42	-	-0.05	-0.42	-0.42
DTM(U)	0.39	-	0.14	0.43	-0.18	-	-0.41	-0.61
DTM(Y)	0.35	0.16	-		-0.24	-0.06	-	-0.37
DTM(W)	0.32	0.17	0.15	-	-0.23	-0.04	-0.44	-
TRAMAR(I)	-	0.12	0.11	0.45	-	-0.08	-0.45	-0.39
TRAMAR(U)	0.64	-	0.10	0.41	0.09	-	-0.47	-0.45
TRAMAR(Y)	0.26	0.10	-	0.42	-0.6	0.09	-	-0.37
TRAMAR(W)	0.35	0.14	0.12	-	-0.15	0.05	-0.42	-
MSL	0.38	0.18	0.16	0.41	-0.17	-0.04	-0.39	-0.63
LSM	**0.71**	**0.21**	0.17	**0.55**	-0.06	**-0.03**	-0.34	**-0.05**

(The abbr: IMDB(I), UW-CSE(U), Yeast(Y), WebKB(W))

Table 4 reports the AUCs and CLLs, averaged over all fold. The abbr are IMDB(I), UW-CSE(U), Yeast(Y), WebKB(W). For instance, the DTM(I) means that we transfer the IMDB(I) to other domains by using algorithm DTM. Specially, DTM(C) means that we combine the SOT from other three domains and transfer them to a target domain.

Clearly, most of transfer learning method yields better results than only structure learning MSL. Since the two search strategies of MSL are both incomplete sometimes (cannot traverse entire candidate clause search space) and limit by the maximum clause length (for reasons of tractability), the search result (FOLs) may not be the optimal solution in some complex domains. But transfer learning method can match the target domain better for it can use declarative bias to reach the search space that MSL may not searched.

The DTM is slightly better than TRMAR in UW-CSE and Yeast datasets, since both DTM and TRMAR are using similar structure of formula learning form source domains. Despite the high time complexity, the CLLs of MMU(MSL) is close to the MMU (LSM) due to the expanded seed network. Our MMU (LSM) is better than other methods that include state-of-the-art MLN structure learning algorithm LSM and MLN-based transfer learning algorithm DTM and TRMAR. It is mainly because our method not only transfer the structure of formula learning form source domains but also transfer new structure of formulas through knowledge connection and mutation. In other words, our method can reach more important search space than other methods.

8. Conclusion and Future Work

We have shown that our MSKI can integrate, hybrid and create new knowledge, which is formalized into an uncertain hypergraph. Then find the frequency sub-hypergraph (pivot knowledge) of the large uncertain hypergraph by using hypergraph DFS code to deal with the hypergraph isomorphism and candidate sub-hypergraphs search space problems. Transfer these pivot knowledge with high priority can reduce the searching space of clause greatly. Our experiments indicate that our method outperforms state-of-the-art single task MLN-based transfer learning. Much more remains to be explored, including expand our algorithm to directed uncertain hypergraph for more accuracy represent of SOLT, design more efficient knowledge integration methods by pruning some knowledge that has less correlation with target domain, and further applications.

Acknowledgements

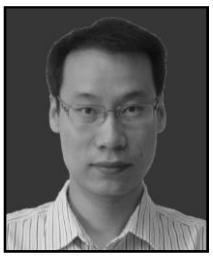
This work is partially supported by the High-Tech Research and Development Plan of China (No. 2012AA012506); the National Natural Science Foundation of China (Grant No. 61173145, 61472108). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers.

References

[1] S. J. Pan and Q. Yang, “A survey on transfer learning”, IEEE Transactions on Knowledge and Data Engineering, vol.22, no.10, **(2010)**.
[2] J. Davis and P. Domingos, “Deep transfer via second-order markov logic”, In Proceedings of the 26th International Conference on Machine Learning. Montreal, **(2009)**; QC, Canada.
[3] H. T. Mihalkova, Lilyana and R. J. Mooney, “Mapping and revising markov logic networks for transfer learning”, In Proceedings of the National Conference on Artificial Intelligence, **(2007)**; Vancouver, BC, Canada.
[4] L. Mihalkova and R. J. Mooney, “Transfer learning by mapping with minimal target data”, In AAAI Workshop - Technical Report, **(2008)**; Chicago, IL, United States.
[5] L. J. G. H. Zou and Zhaonian, “Frequent subgraph pattern mining on uncertain graph data”, In Proceedings of the International Conference on Information and Knowledge Management, **(2009)**; Hong Kong, China.

[6] G. H. Zou, Zhaonian and J. Li, "Discovering frequent subgraphs overuncertain graph databases under probabilistic semantics", In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, **(2010)**; Washington, DC, United States.
[7] B. B. Horvath, Tamas and L. De Raedt, "Frequent hypergraph mining", In ILP:Lecture Notes in Computer Science, **(2007)**, Santiago de Compostela, Spain.
[8] L. J. G. H. Zou and Zhaonian, "Mining frequent subgraph patterns from uncertain graph data", IEEE Transactions on Knowledge and Data Engineering. vol.22, no.9, **(2010)**.
[9] M. Fiedler and C. Borgelt. "Subgraph support in a single large graph", In ICDM Workshops, **(2007)**; Omaha, NE.
[10] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining", In Proceedings of the International Conference on Data Mining, **(2002)**.
[11] S. Kok and P. Domingos, "Learning the structure of markov logic networks", In Proceedings of the 22nd International Conference on Machine Learning, **(2005)**; Bonn, Germany.
[12] M. Richardson and P. Domingos, "Markov logic networks", Machine Learning, vol.62, no.1 **(2006)**.
[13] S. Kok and P. Domingos, "Learning markov logic networks using structural motifs", In Proceedings of the Twenty-Seventh International Conference on Machine Learning, **(2010)**; Haifa, Israel.
[14] D. Moore and A. Danyluk, "Deep transfer as structure learning in markov logic networks", In AAAI-2010 Workshop on Statistical Relational AI, **(2010)**; Atlanta, GA.
[15] S. Kok and P. Domingos, "Learning markov logic network structure via hypergraph lifting", In Proceedings of the International Conference on Machine Learning, **(2009)**; Montreal, QC, Canada.

Authors



Xing Wang, he received the B.S. and M.S. degree in computer science from Northwest University, XiAn. China. He is now working towards his Ph.D. degree in computer science at Harbin Institute of Technology. His research interests include computer network, machine learning, and public opinion.

Bin-Xing Fang, he received his M.S. and Ph.D. degrees in computer science from the Tsinghua University and Harbin Institute of Technology of China in 1984 and 1989 respectively. He is currently a member of Chinese Academy of Engineering. His current research interests include information security, information retrieval, and distributed systems.

Hui He, she received the B.S., M.S. and Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China. Since September 1999, she has been with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, where she became an Associate Professor in October 2007. Her research interests include network computing, network security.

Hong-Li Zhang, she received her M.S. and Ph.D. in Computer Architecture from the Harbin Institute of Technology on July 1996 and December 1999, respectively. Her research interests are focused in the area of network security, Internet measurement and network computing. She was awarded 3 Ministry Science and Technology Progress awards and published over 50 papers in journals and international conferences.