Entity Correspondence with Second-Order Markov Logic

Ying Xu[1,2], Zhiqiang Gao[1], Campbell Wilson[2], Zhizheng Zhang[1], Man Zhu[1], and Qiu Ji[1]

[1] Institute of Computer Science and Engineering, Southeast University, China
[2] Faculty of Information Technology, Monash University, Australia

Abstract. Entity Correspondence seeks to find instances that refer to the same real world entity. Usually, a fixed set of properties exists, for each of which the similarity score is computed to support entity correspondence. However, in a knowledge base that has properties incrementally recognized, we can no longer rely only on the belief that two instances sharing value for the same property are likely to correspond with each other: a pair of different properties that are of hierarchies or specific relations can also be evidential to corresponding instances. This paper proposes the use of second-order Markov Logic to perform entity correspondence. With second-order Markov Logic, we regard properties as variables, explicitly define and exploit relations between properties and enable interaction between entity correspondence and property relation discovery. We also prove that second-order Markov Logic can be rephrased to first-order in practice. Experiments on a real world knowledge base show promising entity correspondence results, particularly in recall.

Keywords: Entity Correspondence, Second-Order Markov Logic, Property Relation Discovery, Knowledge Base.

1 Introduction

Entity correspondence is a task that seeks to find instances that refer to the same real world entity. This task has different names in different fields, e.g. record linkage, entity resolution, instance matching and object reconciliation. In this paper, we focus on entity correspondence in automatically constructed knowledge bases. Different from traditional knowledge bases that rely on experts, these knowledge bases typically consist of probabilistic beliefs that are based on the redundancy of the web information. Several extraction systems, including KnowItAll [1], Never-Ending Language Learner (NELL) [2] and REISA [3], have been developed for this purpose. Many of these systems are based on a bootstrapping approach that ensures automated extraction of instances and properties (including instance categories and relations between instances). NELL, in particular, has accumulated a knowledge base of 1,829,036 asserted instances and 859 different properties after beginning with an initial set of around 800 instances and 188 properties[1]. This knowledge base has a locally-formatted ontology as its core for supporting information extraction, and in return, automatically extracted information can be used to enrich the knowledge base. Although it is inspiring that a machine can read the web and acquire knowledge like a human, the resulting knowledge base cannot be linked to other datasets that are available on the web, limiting the use of such a valuable knowledge base. The Semantic Web[2] is proposed to allow data to be shared and reused across application, enterprise, and community boundaries. We aim to link a knowledge base like NELL into the Semantic Web, and focus on entity correspondence as the first step to do so.

Several features of automatically constructed knowledge bases make the entity correspondence different from previous efforts. 1) The properties are incrementally recognized and thus infinite, in which case, we cannot write a fixed set of rules to correspond instances. 2) The machine may recognize properties that are of hierarchies or specific relations, which leads to hidden corresponding instance pairs. For example, in Fig. 1, if we don't know the relation between *HasWife* and *HasSpouse*, we will lose a potential corresponding pair *SameAs*(*Ann1*, *Ann2*). 3) Due to feature 1), we cannot provide fixed property relations for entity correspondence, which requires discovering such relations at the same time with entity correspondence. Although we have the property relation *SubPropertyOf*(*HasWife*, *HasSpouse*) in Fig. 1, we may still lose the potential corresponding pair *SameAs*(*Bob1*, *Bob2*) if we cannot discover the relation between *AgentCreate* and *BookWriter*, which are newly recognized. Notice that discovering such relations and corresponding instances are in fact interactive processes, and simultaneously performing these two can help us to find more corresponding instance pairs. The Semantic Web provides the standard for relations between properties as property hierarchies *SubPropertyOf*[4], which states that all resources related by one property are also related by another, and one of the property characteristics *InverseOf*[4], which states that one property can be obtained by taking another property and changing its direction. Discovering these two kinds of relations between properties not only better supports the task of entity correspondence, but also enables us to organize the automatically recognized properties in accordance with the standard of the Semantic Web.

In this paper, we propose the use of second-order Markov Logic [4] to do entity correspondence with the help of property relation discovery. We solve the issue of incremental properties by regarding properties as variables that can be instantiated with any recognized property, which is naturally enabled by second-order Markov Logic; we solve the issue of hidden property relations by explicitly defining relations between properties using second-order logic constructors and provide discovered corresponding instances pairs as evidence for discovering property relations. And then we show how the interactive process in Fig. 1 is enabled and how entity correspondence is improved in Markov Logic. In addition, we prove that second-order Markov Logic can be rephrased to first-order in practice, which bridges these two models theoretically.

[1] `http://rtw.ml.cmu.edu/rtw/`

[2] `http://www.w3.org/2001/sw/`

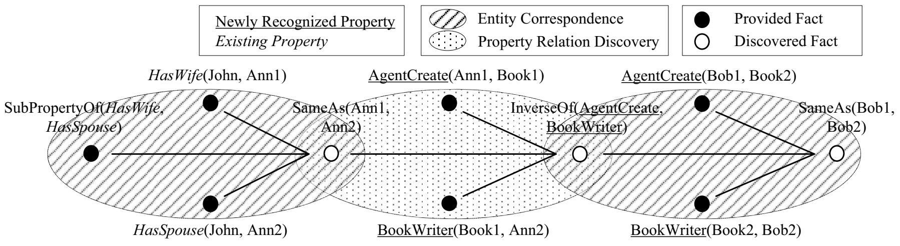

Fig. 1. Interaction between entity correspondence and property relation discovery

2 Preliminary

2.1 First-Order Markov Logic

First-order logic formulae are constructed using four types of symbols: constants, variables, functions and predicates. Constants represent objects in the domain of interests. Variable ranges over objects. Function maps tuples of objects to objects. Predicates represent relations among objects. A *term* can be any constant, variable or any function applied to a tuple of terms. An *atom* (or *atomic formula*) is a predicate symbol applied to a tuple of terms and *formulae* are recursively constructed from atoms using logical connectives and quantifiers, e.g. $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence), $\forall$ (universal quantification) and $\exists$ (existential quantification). Grounding of formulae means replacing variables with constant symbols [5].

Markov network defines the joint distribution of a set of random variables $\boldsymbol{X} = \{X_1, X_2, ..., X_n\} \in \mathcal{X}$. It is composed of an undirected graph $\mathcal{G}$ and a set of potential functions ϕ_k. The graph has a node for each random variable, and a potential function for each *clique*[3]. Usually, the potential functions are conveniently represented in log-linear form, where the potential function is replaced by an exponentiated weighted sum of features of a state. The probability of a state $\boldsymbol{x}$ is then given by $P(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z}\exp(\sum_{j=1}^{N} \omega_j f_j(\boldsymbol{x}))$. Here, f_j is the feature function for jth clique, Z is a normalizing constant defined as $Z = \sum_{\boldsymbol{x}\in\mathcal{X}} \exp(\sum_{j=1}^{N} \omega_j f_j(\boldsymbol{x}))$ and N is the number of cliques [5].

Definition 1. *A Markov Logic Network (MLN) [5] L is a set of pairs (F_i, ω_i), where F_i is a formula in first-order logic and ω_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, ..., c_{|C|}\}$, it defines a Markov network $M_{L,C}$ as follows:*

1. $M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true and 0 otherwise.

[3] If every two nodes in the node set $\mathcal{X}$ are connected by an edge, we call $\mathcal{X}$ a clique [6].

2. $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true and 0 otherwise. The weight of the feature is the ω_i associated with F_i in L.

The MLNs work as templates for constructing Markov networks. Each of these networks is called a *ground Markov network*, where each clique corresponds to a ground formula. The probability distribution over state $\boldsymbol{x}$ is as follows.

$$P(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z} \exp(\sum_i \omega_i n_i(\boldsymbol{x})) \tag{1}$$

Here, n_i is the number of true groundings of F_i in state $\boldsymbol{x}$.

2.2 Second-Order Markov Logic

Extending first-order Markov Logic to second-order involves grounding atoms with all possible predicate symbols as well as all constant symbols. It was first introduced by Kok and Domingos in order to perform statistical predicate invention [4]. Subsequently, Davis and Domingos reused this model to enable deep transfer among different domains [7]. Second-order Markov Logic is a probabilistic extension to second-order logic, which allows quantifications of predicates by quantifiers. For example, $\forall p \forall x (p(x) \vee \neg p(x))$ states that for each unary relation (or predicate) p of individuals and each individual x, either x is in p or it is not. This reflects a common pattern that can be shared by all unary predicates.

3 Bridge First-Order and Second-Order Markov Logic

In order to facilitate the prove of our theorem, we introduce a more general representation of higher order logic, which is called *Relational Type Theory* [8].

Definition 2 ([8]). *Types(or* relational types*) are defined as syntactic expressions inductively generated by: ι is a type; and if $\tau_1 ... \tau_k$ are types ($k \geq 0$), then $\tau = (\tau_1 ... \tau_k)$ is a type.*

In this paper, we intend ι to denote the type of individuals, and $(\iota_1 ... \iota_k)$ the type of k-ary relations between k objects of types $\iota_1 ... \iota_k$ respectively.

Definition 3 ([8]). *Let V be a vocabulary for the second-order formulae. A* ***Henkin-V-prestructure*** *$\mathcal{H}$ consists of*

- *a non-empty universe A;*
- *an interpretation in A of the V-constants; and*
- *for each type τ, a collection D^{τ}, where $D^{\iota} = A$, and $D^{(\tau_1 \cdots \tau_k)} \subseteq \mathcal{P}(D^{\tau_1} \times ... \times D^{\tau_k})$*

Example 1. For a second order formula that consists of unary atom $p(x)$, 2-ary atom of individuals $r(x, y)$, and 2-ary atom of predicates $InverseOf(p_1, p_2)$, the vocabulary V is formalized as $V = \{x, y, p, r, p_1, p_2, InverseOf\}$. According to Definition 2, x and y are of type ι; p is of type (ι); r, p_1 and p_2 are of type $(\tau_1 \tau_2)$; and *InverseOf* is a V-constant of type $(\tau_1 \tau_2)$ where $\tau_1 = (\tau_{11} \tau_{12})$ and $\tau_2 = (\tau_{21} \tau_{22})$. Thus, we can construct a *Henkin-V-prestructure* $\mathcal{H}$ where

- A denotes the universe;
- the V-constant *InverseOf* is interpreted as $InverseOf(A_{(\tau_{11}\tau_{12})}, A_{(\tau_{21}\tau_{22})})$;
- and the collection A is for type ι, collection $D^{(\iota)}$ for type (ι) and collection $D^{(\tau_1\tau_2)}$ for type $(\tau_1\tau_2)$.

Let V^S be an extension of V with relation-constants $T_\iota, T_{(\iota)}, T_{(\tau_1\tau_2)}, BelongsTo$ and $Triple$, of arities 1, 1, 1, 2 and 3 respectively. $T_\iota(x)$ is intended to state x is an individual, $T_{(\iota)}(x) - x$ is a set, $T_{(\tau_1\tau_2)}(x) - x$ is a 2-ary predicate, $BelongsTo(x, p)$ – x is an element of p and $Triple(x, r, y) - x$ and y have relation r.

Definition 4. *A Henkin-V-prestructure $\mathcal{H}$ can determine a unique V^S-**structure** $\mathcal{S} = \mathcal{S}(\mathcal{H})$ where*

- $A^S = A \cup D^{(\iota)} \cup D^{(\tau_1\tau_2)}$;
- *the interpretation in S of the V-constant, i.e. InverseOf in Example 1, is the same as their interpretation in $\mathcal{H}$; and*
- *the V^S-constant T_ι is interpreted as A, $T_{(\iota)}$ as $D^{(\iota)}$, $T_{(\tau_1\tau_2)}$ as $D^{(\tau_1\tau_2)}$, BelongsTo as $A \times D^{(\iota)}$, and Triple as $A \times A \times D^{(\tau_1\tau_2)}$*

For example 1, the second-order V-formula φ *is rephrased to* a first-order V^S-formula φ^S, which is obtained by: replacing each atom $p(x)$ with $BelongsTo(x, p)$ and $r(x, y)$ with $Triple(x, r, y)$; relativizing quantifiers over individuals to T_ι, quantifiers over sets to $T_{(\iota)}$, and quantifiers over 2-ary predicates to $T_{(\tau_1\tau_2)}$.

Lemma 1 ([8]). *A formula φ is true in $\mathcal{H}$ iff φ^S is true in $\mathcal{S}(\mathcal{H})$.*

According to Lemma 1, we can conclude the following theorem.

Theorem 1. *A second-order MLN L can determine a first-order MLN L^S that have the same probability distribution by replacing unary atom $p(x)$ with Belongs To(x, p) and 2-ary atom $r(x, y)$ with $Triple(x, r, y)$.*

Proof. Let $F = \{F_i | i = 0, ..., N\}$ be the set of formulae in L and $F^S = \{F_i^S | i = 0, ..., M\}$ the set of formulae in L^S. Each F_i^S is obtained by replacing unary atom $p(x)$ with $BelongsTo(x, p)$ and 2-ary atom $r(x, y)$ with $Triple(x, r, y)$ in F_i. So, we have $N = M$. Let V and V^S be the vocabulary for F and F^S respectively. We can construct a Henkin-V-prestructure $\mathcal{H}$ for F which uniquely determines a V^S-structure $\mathcal{S}(\mathcal{H})$. According to Lemma 1, F_i in $\mathcal{H}$ and F_i^S in $\mathcal{S}(\mathcal{H})$ have the same truth value.

Let n_i be the number of true groundings of the ith formula F_i in L, and n_i^S the number of true groundings of F_i^S in L^S, we have $n_i = n_i^S$, which leads to $P(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z}\exp(\sum_{i=1}^{N} \omega_i n_i(\boldsymbol{x})) = \frac{1}{Z}\exp(\sum_{i=1}^{M} \omega_i n_i^S(\boldsymbol{x}))$.

4 Entity Correspondence

4.1 Basic Model

Human beings have knowledge about how to correspond instances using the properties they have. For example, it's intuitive to us that persons that are married to the same person *are likely to* refer to the same real world entity. Such knowledge can be regarded as a rule that is not certainly but probably correct, which can be perfectly modeled by a weighted formula in MLN. Accordingly, properties are mapped to predicates and property relations are mapped to predicates that take predicates as arguments. We introduce the predicate *SameAs* to correspond two instances. Then, for the above example, we can write a weighted formula like the following,

$$\begin{aligned}(\forall x_1, x_2, x_3 \quad &(\text{MarriedTo}(x_1, x_3) \ \wedge \ \text{MarriedTo}(x_2, x_3) \\ &\rightarrow \ \text{SameAs}(x_1, x_2)), \quad \omega)\end{aligned} \tag{2}$$

where ω illustrates how strong the formula is: higher weight means greater difference between a state that satisfies the formula and the one that does not.

In an automatically constructed knowledge base, properties are recognized now and then, which results in incremental number of properties. Thanks to second-order Markov Logic, we can regard predicates as variables as well, which leads to more general formulae that can be instantiated with any predicate symbol. Formula (3) and (4) are our second-order formulae for entity correspondence, where we replace the first and second predicate-constant *MarriedTo* in Formula (2) with predicate variable p_1 and p_2, considering that a pair of different properties can be evidential to entity correspondence. And in order to provide relations between properties as evidence, we introduce predicates *CloseTo* and *InverseOf* for *SubPropertyOf* and *InverseOf* that are mentioned in Sect. 1. For the predicate *CloseTo*, we remove the constraint of *SubPropertyOf* that the direction cannot be reversed. For example, *SubPropertyOf*(*HasSpouse*, *HasWife*) has truth value *false* while *CloseTo*(*HasSpouse*, *HasWife*) has *true*. We enable this compromise because both directions of *SubPropertyOf* have the same effect on entity correspondence.

$$\begin{aligned}(\forall p_1, p_2 \quad \forall x_1, x_2, x_3 \quad &\text{Similar}(x_1, x_2) \ \wedge \ p_1(x_1, x_3) \ \wedge \ p_2(x_2, x_3) \\ \wedge \ &\text{CloseTo}(p_1, p_2) \ \rightarrow \ \text{SameAs}(x_1, x_2)), \quad \omega_1)\end{aligned} \tag{3}$$

$$\begin{aligned}(\forall p_1, p_2 \quad \forall x_1, x_2, x_3 \quad &\text{Similar}(x_1, x_2) \ \wedge \ p_1(x_1, x_3) \ \wedge \ p_2(x_3, x_2) \\ \wedge \ &\text{InverseOf}(p_1, p_2) \ \rightarrow \ \text{SameAs}(x_1, x_2)), \quad \omega_2)\end{aligned} \tag{4}$$

These two formulae are relatively aggressive if placed in a rule-based model as we cannot correspond instances using only one pair of properties. Some properties, e.g. *WorkFor*, maybe too weak to correspond two instances, where we need more evidence. However, from the point of view of Markov Logic, they make sense. At first, Formula (3) and (4) are just templates for generating Markov networks, where each formula may be instantiated to multiple ground formulae. More shared properties means more true ground formulae, and thus higher probability of a state. Secondly, the weighted formula itself indicates that the rule is partially correct, where each true ground formula just improves the probability of a state, but not make a certain decision. In addition, the whole Markov network generated is a connected one, where the probability of each node is influenced by many other factors, which makes the property *WorkFor* not so aggressive in corresponding entities.

Then, how can we discover property relations? Reviewing the example given in Fig. 1, we can in fact easily conclude the pattern for property relation discovery as following two weighted formulae:

$$\begin{aligned}&(\forall p_1, p_2 \quad \forall x_1, x_2, x_3 \quad (\mathrm{SameAs}(x_1, x_2) \ \wedge \ p_1(x_1, x_3) \ \wedge \ p_2(x_2, x_3) \\ &\rightarrow \ \mathrm{CloseTo}(p_1, p_2)), \quad \omega_3)\end{aligned} \tag{5}$$

$$\begin{aligned}&(\forall p_1, p_2 \quad \forall x_1, x_2, x_3 \quad (\mathrm{SameAs}(x_1, x_2) \ \wedge \ p_1(x_1, x_3) \ \wedge \ p_2(x_3, x_2) \\ &\rightarrow \ \mathrm{InverseOf}(p_1, p_2)), \quad \omega_4)\end{aligned} \tag{6}$$

In order to keep accordance with the symmetry and transitivity characteristics of the predicate *SameAs*, we also define symmetric and transitive *hard constraints* for *SameAs*. Additionally, *CloseTo* and *InverseOf* also have the symmetry characteristic that can refine the results of property relation discovery.

Existing tools for Markov Logic include Alchemy[4] and Tuffy[5], which are well implemented and maintained. However, none of them support second-order Markov Logic. In order to reuse these tools, we reformulate Formula (3) to (6) to first-order according to Theorem 1, replacing all 2-ary atoms $p(x, y)$ with $Triple(x, p, y)$. In the rest of this paper, we intend Formula (3)' to(6)' to denote the first-order form of Formula (3) to (6).

4.2 Decreasing the Number of Candidate Corresponding Pairs

The basic idea of *canopy* [9] is to find some measure to separate candidates that are obviously unmatched so as to decrease the number of candidate corresponding pairs. We use two measures, *name similarities* and *instance categories*, to decrease the number of candidate corresponding pairs.

1. We *only* compare instances with similar name labels. The similarity score is computed by the Overlap Coefficient[11], $\mathrm{Overlap}(X, Y) = \frac{|X \cap Y|}{\min(|X|, |Y|)}$, where X and Y are two string sets. We introduce the predicate *Similar* for this measure and provide it as an atom of the conjunctive antecedents of Formulae (3)' and (4)'. Thus, only similar names can activate them.
2. We *only* compare instances that are in the same category. Considering the case of entity correspondence within one category, we divide the collection of instances into two subsets, one for the target category, and one for the rest. As a result, the predicate *Triple* should be replaced by *Triple*$_1$ and *Triple*$_2$, which indicates that the target instance is the first and second argument respectively. Corresponding changes should be taken to other predicates.

4.3 Weight Learning

Up to this point, we have introduced the model for entity correspondence and property relation discovery. In practice, weights can be specified manually or

[4] `http://alchemy.cs.washington.edu/`
[5] `http://hazy.cs.wisc.edu/hazy/tuffy/`

learnt from training data. We use discriminative learning where conditional probability of query atoms are computed given the evidence nodes. According to Equation (1), the partial derivation of the conditional log-likelihood of Markov networks is given by $\frac{\partial}{\partial \omega_i} \log P_\omega(\boldsymbol{y}|\boldsymbol{x}) = n_i(\boldsymbol{x}, \boldsymbol{y}) - E_\omega[n_i(\boldsymbol{x}, \boldsymbol{y})]$, where $\boldsymbol{x}$ is the vector of the evidence atoms' truth values, $\boldsymbol{y}$ is the truth values of the query atoms and $n_i(\boldsymbol{x}, \boldsymbol{y})$ is the number of true groundings of ith formula [12]. Thus, we can use the standard gradient-based optimization methods to learn weights.

4.4 Inference

Inference allows for query about ground atoms of one or more predicates. We aim to capture dependencies between entity correspondence and property relation discovery, so we query *SameAs*, *CloseTo* and *InverseOf* at the same time. The most widely used algorithms for inference in Markov Logic include (Loopy) Belief Propagation, Simulated Tempering, MC-SAT and Gibbs Sampling. As we have circles in our model, loopy belief propagation may not converge at last. Simulated tempering is much slower than the others as it includes a process of slow cooling to extend the solution space. According to our experiments on small datasets, Gibbs Sampling is relatively faster than MC-SAT, while, at the same time, gives equally satisfying results. So we choose Gibbs Sampling for the conditional probabilistic query. This kind of inference tells us how likely two instances correspond with each other given the evidences and we can set different thresholds to filter the results.

4.5 An Example of Bi-directional Joint Inference

In this section, we demonstrate how entity correspondence is improved in Markov Logic, taking property relation discovery into consideration. Reviewing Fig. 1, if we query the facts at white nodes and provide the black ones as evidence, Fig. 2 illustrates the Markov network that is generated after grounding. Notice that *Similar* atoms are provided additionally and instance relations are rephrase to *Triple* atoms. Here, Fig. 2 (a) is a clique that corresponds to Formula (3)', Fig. 2 (b) is a clique that corresponds to Formula (6)' and Fig. 2 (c) is a clique that corresponds to Formula (4)'. Given the evidence nodes in Fig. 2 (a), the query node *SameAs*(*Ann1*, *Ann2*) has high probability to be assigned true, which results in high probability of node *InverseOf*(*AgentCreate*, *BookWriter*) in Fig. 2 (b) and further affects the probability of node *SameAs*(*Bob1*, *Bob2*) in Fig. 2 (c). Fig. 2 (d) demonstrates the directions of message propagation between entity correspondence and property relation discovery. Fig. 2 (a) and Fig. 2 (b) propagate massage from entity correspondence to property relation discovery, which we call *single-directional* message propagation, and Fig. 2 (b) and Fig. 2 (c) propagate message from property relation discovery to entity correspondence in return, which completes the *bi-directional* message propagation loop. Notice that the above process is completed within one query, and the assignment to each query node will be optimized in a global view. As a result, entity correspondence and property relation discovery are collectively improved.

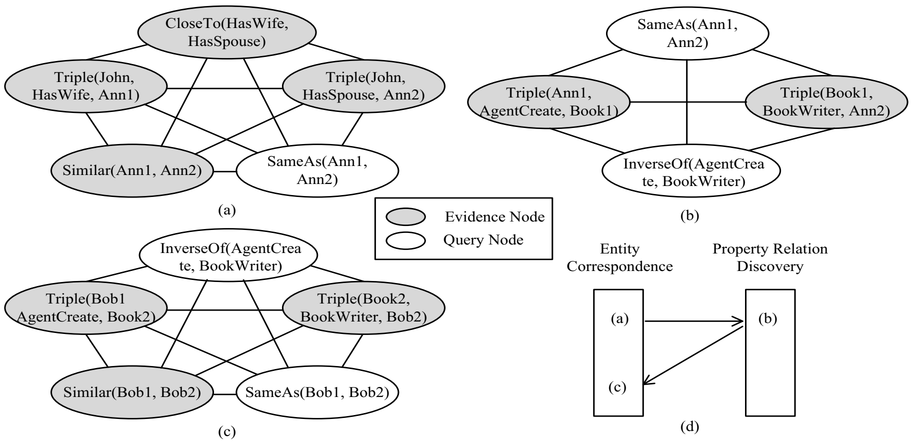
CloseTo(HasWife, HasSpouse)
Triple(John, HasWife, Ann1)
Triple(John, HasSpouse, Ann2)
Similar(Ann1, Ann2)
SameAs(Ann1, Ann2)
(a)
SameAs(Ann1, Ann2)
Triple(Ann1, AgentCreate, Book1)
Triple(Book1, BookWriter, Ann2)
InverseOf(AgentCreate, BookWriter)
(b)
Evidence Node
Query Node
InverseOf(AgentCreate, BookWriter)
Triple(Bob1 AgentCreate, Book2)
Triple(Book2, BookWriter, Bob2)
Similar(Bob1, Bob2)
SameAs(Bob1, Bob2)
(c)
Entity Correspondence
Property Relation Discovery
(a)
(b)
(c)
(d)

Fig. 2. Message propagation between entity correspondence and property relation discovery

5 Experiments and Analysis

5.1 Dataset

The dataset we use for experiments is the iteration 690 of the NELL knowledge base[6]. This downloadable part of NELL knowledge base contains 1,850,160 beliefs in total, among which 1,795,281 have high confidence greater than 0.9. For experiments, we focus on entity correspondence within person category. So we get 285,793 person instances and extracts those have at least one relation with other instances. The resulting candidate instances we can use is only 10,550 of them. We manually annotate the corresponding pairs among the 10,550 instances, and randomly choose 300 of them that have at least one *SameAs* relation with others. The final *SameAs* matrix is of size 300×300, and the number of true entries in the *SameAs* matrix is 566. Notice that grounding involves replacing all variables with all possible constants, so the generated Markov networks for the above 300 instances contain tens of thousands of nodes in practice, which limits the scale of the datasets. Nevertheless, we build a benchmark that can be used to evaluate entity correspondence with automatically recognized properties.

5.2 Experiment Settings

We conduct four groups of experiments to demonstrate four different aspects of our model. At first, We incrementally extend the original model to enable it to perform entity correspondence and property relation discovery collectively. The five models given in Table 1 are all based on Markov Logic and follow a similar

[6] `http://rtw.ml.cmu.edu/resources/results/08m/NELL.08m.690.esv.csv.gz`

Table 1. Five Markov Logic models for entity correspondence

Name	Discription
MLN+Orig	Extract properties in the training set, for each of which formula like Formula (2) is manually written.
MLN+PP	Write formulae for each pair of properties in the training set, e.g. $(\forall x_1, x_2, x_3(\text{Similar}(x_1, x_2) \wedge \text{HasSpouse}(x_1, x_3) \wedge \text{HasWife}(x_2, x_3) \rightarrow \text{SameAs}(x_1, x_2), \quad \omega)$
MLN+PR	Include predicate *InverseOf* and *CloseTo*; add Formulae (5)' and (6)'; enables *single-direction* message propagation.
MLN+PR+PV	Regard properties as variables; enables *bi-direction* message propagation. (This is the model described in Sect. 4.)
MLN+PRH+PV	Define Formulae (5)' and (6)' as hard.

nomenclature to that used in [10]. For this group, we aim to show the difference after taking property relation discovery into consideration. Secondly, we compare our approach with first-order Markov Logic [10] and the Fellegi-Sunter pairwise approach [13]. The first-order Markov Logic approach is exactly the MLN+Orig mentioned above. The Fellegi-Sunter approach is the very original pairwise model, which regards entity correspondence as a classification problem where a vector of similarity scores is given as feature. For this group, we aim to show the difference between pairwise approach and collective approach. Thirdly, as we already have property relation in training set before inference, we compare situations where we provide these property relations as evidence for test and not do so. Fourthly, we compare among MLN+PR, MLN+PR+PV and MLN+PRH+PV at discovering property relations. The correctness of these discovered property relations are checked manually.

We perform 5-fold cross validation to the whole dataset. For each iteration, we use four folds for learning and one for test. And we perform precision, recall and F1 evaluations against the benchmark we build for entity correspondence.

5.3 Experimental Results

In Figure 3, we have two thresholds. The similarity threshold is used to decide the truth value of *Similar* atoms, which are assigned *true* if the similarity scores are greater than the threshold. The probability threshold is used to decide how probable two instances correspond with each other should they be considered a positive example of corresponding instances. There is no obvious difference between the results of similarity threshold 0.5 and 0. However, the precision in Fig. 3 (a) does have a little improvement compared to that in Fig. 3 (d), which is to our intuition that higher similarity threshold can keep out more noise, and thus gives higher precision. The results within each subgraph show that with the increase of probability threshold, precisions increase and recalls decrease.

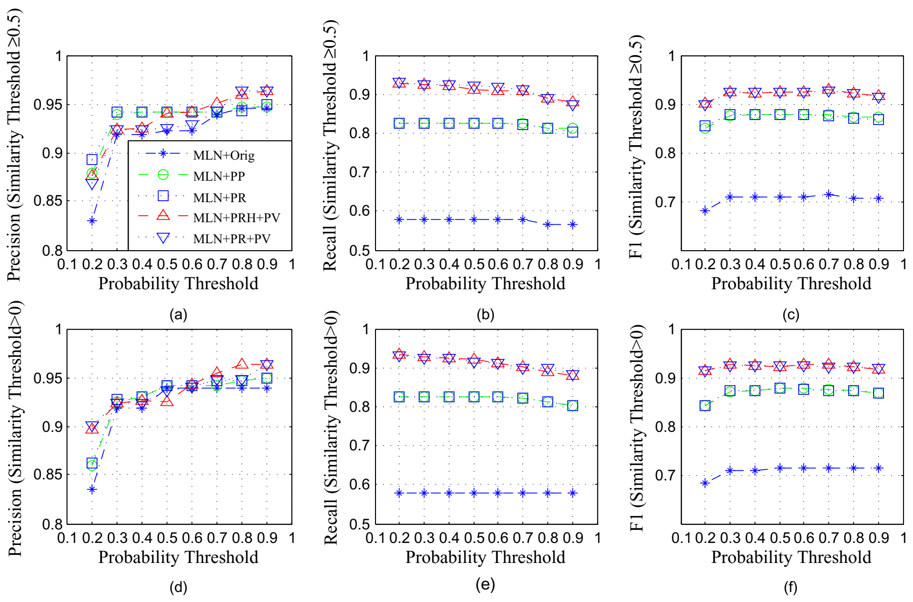

Fig. 3. Comparison between different Markov Logic models for entity correspondence

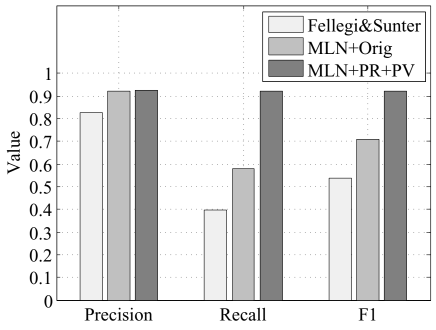

Fig. 4. Comparison between different approaches for entity correspondence (probability threshold=0.5)

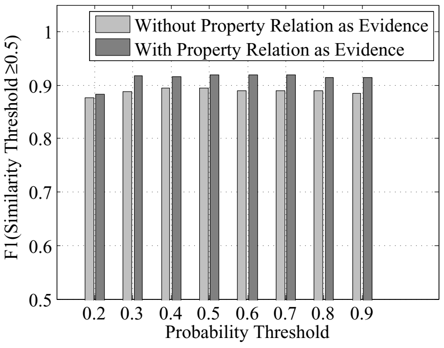

Fig. 5. Entity correspondence before and after providing partial property relations as evidence in MLN+PRH+PV

Table 2. Results for property relation discovery

	Sim. Threshold ≥ 0.5		Sim. Threshold > 0	
	Avg. Prec.	# Correct	Avg. Prec.	# Correct
MLN+PR	90.39	53	88.06	54
MLN+PR+PV	90.60	**62**	**90.43**	**60**
MLN+PRH+PV	**90.77**	60	87.92	60

Figure 3 demonstrates that our models (MLN+PRH+PV and MLN+PR+PV) have better precisions (96.29%) than the others (94.90%) when the probability threshold is greater than 0.5. More obvious improvement is in recall, which is to our expectation that after taking property relation discovery into consideration, we can find more corresponding instances. MLN+Orig has the worst recalls as it loses a lot of potential corresponding pairs that are evidenced by related properties. MLN+PP and MLN+PR have almost the same results, except that MLN+PR can be used to discover new property relations. However, the newly discovered property relations will not affect entity correspondence in return as the entity correspondence formulae are fixed by pre-specified properties. In our models, bi-directional message propagation is enabled by defining properties as variables, increasing recall from 82.38% to 92.22% (probability threshold=0.5).

Figure 4 demonstrates the results for Fellegi-Sunter, MLN+Orig and MLN+PR+PV. Their precisions are 82.67%, 92.20% and 92.51%, and their recalls are 39.91%, 57.97% and 92.22% respectively. These obvious improvements come from the limitation of pairwise approaches and the first-order Markov Logic. In datasets with complex relations, pairwise approaches may lose many valuable dependencies among multiple entity correspondence decisions, for example, $SameAs(A,B)$ and $SameAs(B,C)$ can lead to $SameAs(A,C)$. First-order MLN captures such dependencies, but does not capture dependencies between entity correspondence and property relation discovery, as it is not capable of defining relations between properties. Our model enables both of the above dependencies, so we have the best results for entity correspondence. However, as the Markov network inference is #P-complete [5], we would expect much longer processing time than using pairwise model. For our problem, the time for MLN inference varies from 48 to 80 minutes, while the pairwise approach only take 4 seconds for testing. Speeding up MLN inference is an interesting problem, which should be a good point of view for our future works.

Figure 5 shows that F1 have a general increase if we provide partial property relations as evidence in MLN+PRH+PV. This is because evidence nodes have higher probability than query nodes and thus can better support entity correspondence. However, as the properties are incrementally recognized, we can never provide relations between newly recognized properties. This is why we need to discovery these relations.

Table 2 demonstrates the results for property relation discovery. The results show that MLN+PR+PV and MLN+PRH+PV can discover more related properties, which, we believe, is the result of their higher recalls for entity correspondence. It implies that dependencies between entity correspondence and property relations discovery can improve the results of both tasks.

6 Related Works

Entity Correspondence has been received a wide range of research in different fields. It is regarded as a classification problem [13] where a pair of candidates is classified as 'Match' or 'Not Match' given a vector of similarity scores. However, recent works focus on capturing dependencies among multiple decisions.

For example, Singla and Domingos[10], Parag and Domingos [14] try to model dependencies among paper correspondence and author correspondence for citation matching, as one instance may appear in multiple candidate matching pairs. Brocheler [15] propose Probabilistic Similarity Logic for reasoning about entity resolution. Bhattacharya and Getoor [16] propose collective entity resolution to jointly determine corresponding entities for co-occuring references. More recent works include joint entity resolution [17], whereby the result of resolving one dataset may benefit the resolution of another dataset.

Another kind of dependency comes from entity correspondence and other tasks. Poon and Domingos [18], Singh [19] allow dependencies between entity correspondence and segmentation, which improves the result for both tasks. Haas [20] perform data integration by leveraging information about both schema and data to improve the integrated results. Niepert [21] tries to map two ontologies by using schema information to exclude logically inconsistent corresponding pairs. Whang [22] does not explicitly capture dependencies among multiple tasks, but it tries to adapt entity correspondence rules to the changing data, where rule refinement and entity correspondence are collectively performed. Our work tries to capture the dependency between entity correspondence and property relation discovery, which is enabled by bi-directional joint inference in Markov Logic.

We use the same graphical model with [10], but we extend the first-order Markov Logic to second-order to enable discovering property realtions. We focus on automatically constructed knowledge bases that are full of probabilistic facts, while in [20][17][22], they have reliable data sources saved in databases. We have to discover relations between properties within the same knowledge base, while in [21], there are well-defined ontologies on both sides. In addition, as we regard property as variable, our model can be applied to any set of properties.

7 Conclusions

In this paper, we propose to perform entity correspondence with second-order Markov Logic, which enables us to support entity correspondence by property relation discovery. As evidenced by the experimental results, these two tasks can strengthen each other when joint inference of Markov Logic is performed. We compare among five different models, extending incrementally from the original model to our models, demonstrating consequent improvements in both precision and recall for entity correspondence. We also compare among three different approaches to illustrate the importance of capturing dependencies among multiple decisions and tasks. In addition, results for property relation discovery show an improvement when entity correspondence is improved.

Acknowledgement. We gratefully acknowledge funding from the National Science Foundation of China under grants 61170165.

References

1. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open Information Extraction from the Web. Communications of the ACM 51(12), 68–74 (2008)

2. Carlson, A., Betteridge, J., Kisiel, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an Architecture for Never-Ending Language Learning. In: 24th AAAI, vol. 2(4), pp. 1306–1313 (2010)
3. Mrabet, Y., Bennacer, N., Pernelle, N.: Controlled Knowledge Base Enrichment from Web Documents. In: Wang, X.S., Cruz, I., Delis, A., Huang, G. (eds.) WISE 2012. LNCS, vol. 7651, pp. 312–325. Springer, Heidelberg (2012)
4. Kok, S., Domingos, P.: Statistical Predicate Invention. In: 24th Annual International Conference on Machine Learning, pp. 433–440. ACM (2007)
5. Richardson, M., Domingos, P.: Markov Logic Networks. Machine Learning 62(1-2), 107–136 (2006)
6. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
7. Davis, J., Domingos, P.: Deep Transfer via Second-order Markov logic. In: 26th Annual International Conference on Machine Learning, pp. 217–224. ACM (2009)
8. Leivant, D.: Higher Order Logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, pp. 229–321 (1994)
9. McCallum, A., Nigam, K., Ungar, L.H.: Efficient Clustering of High-dimensional Data Sets with Application to Reference Matching. In: 6th ACM SIGKDD, pp. 169–178. ACM (2000)
10. Singla, P., Domingos, P.: Entity Resolution with Markov Logic. In: 6th International Conference on Data Mining, pp. 572–582. IEEE (2006)
11. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)
12. Singla, P., Domingos, P.: Discriminative Training of Markov Logic Networks. In: 20th AAAI, vol. 5, pp. 868–873. AAAI Press (2005)
13. Fellegi, I.P., Sunter, A.B.: A Theory for Record Linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969)
14. Domingos, P.: Multi-Relational Record Linkage. In: Proceedings of the KDD 2004 Workshop on Multi-Relational Data Mining, pp. 31–48 (2004)
15. Brocheler, M., Mihalkova, L., Getoor, L.: Probabilistic Similarity Logic. Technical report, University of Maryland, College Park (2010)
16. Bhattacharya, I., Getoor, L.: Collective Entity Resolution in Relational Data. TKDD 1(1), 1–35 (2007)
17. Whang, S.E., Garcia-Molina, H.: Joint Entity Resolution. In: 28th International Conference on Data Engineering (ICDE). IEEE (2012)
18. Poon, H., Domingos, P.: Joint Inference in Information Extraction. In: Proceedings of the National Conference on Artificial Intelligence (2007)
19. Singh, S., Schultz, K., McCallum, A.: Bi-directional Joint Inference for Entity Resolution and Segmentation using Imperatively-defined Factor Graphs. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 414–429. Springer, Heidelberg (2009)
20. Haas, L.M., Hentschel, M., Kossmann, D., Miller, R.J.: Schema and Data: A Holistic Approach to Mapping, Resolution and Fusion in Information Integration. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 27–40. Springer, Heidelberg (2009)
21. Niepert, M., Noessner, J., Meilicke, C., Stuckenschmidt, H.: Probabilistic-Logical Web Data Integration. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 504–533. Springer, Heidelberg (2011)
22. Whang, S.E., Garcia-Molina, H.: Entity Resolution with Evolving Rules. Proceedings of the VLDB Endowment 3(1-2), 1326–1337 (2010)