Constrained Information-Theoretic Tripartite Graph Clustering to Identify Semantically Similar Relations

Chenguang Wang[a], Yangqiu Song[b], Dan Roth[b], Chi Wang[c], Jiawei Han[b], Heng Ji[d], Ming Zhang[a]
[a]School of EECS, Peking University
[b]Department of Computer Science, University of Illinois at Urbana-Champaign
[c]Microsoft Research, [d]Department of Computer Science, Rensselaer Polytechnic Institute
wangchenguang@pku.edu.cn, {yqsong,danr}@illinois.edu, chiw@microsoft.com
hanj@illinois.edu, jih@rpi.edu, mzhang@net.pku.edu.cn

Abstract

In knowledge bases or information extraction results, differently expressed relations can be semantically similar (e.g., (X, wrote, Y) and (X, 's written work, Y)). Therefore, grouping semantically similar relations into clusters would facilitate and improve many applications, including knowledge base completion, information extraction, information retrieval, and more. This paper formulates relation clustering as a constrained tripartite graph clustering problem, presents an efficient clustering algorithm and exhibits the advantage of the constrained framework. We introduce several ways that provide side information via must-link and cannot-link constraints to improve the clustering results. Different from traditional semi-supervised learning approaches, we propose to use the similarity of relation expressions and the knowledge of entity types to automatically construct the constraints for the algorithm. We show improved relation clustering results on two datasets extracted from human annotated knowledge base (i.e., Freebase) and open information extraction results (i.e., ReVerb data).

Introduction

A relation triplet (e^1, r, e^2) is one popular form for knowledge representation. For example, in a knowledge base, such as Freebase[1], a typical relation triplet contains $e^1 = \text{Larry Page}$, $e^2 = \text{Google}$, and $r = \text{is founder of}$. This means that two entities "Larry Page" and "Google" hold the relation "is founder of." With the recent development of knowledge graph and open information extraction (open IE) [Banko *et al.*, 2007; Fader *et al.*, 2011; Schmitz *et al.*, 2012], there are many cases where multiple relation expressions indicate semantically similar relations.[2] The ability to group semantically similar relations into clusters would facilitate and improve many applications, including knowledge base completion, information extraction, information retrieval, and more. Consider the following examples.

Ex. 1: *(X, wrote, Y)* and *(X, 's written work, Y)*.

These two relations are identical, since the meaning of the two expressions is the same when X and Y are instantiated. This kind of relation clustering is very useful for predicate invention [Kok and Domingos, 2007] and knowledge base completion [Socher *et al.*, 2013; West *et al.*, 2014], since we can easily replace the entities (e.g., X or Y) of one relation with the corresponding entities of another one, and use different relation expressions to search for more entities.

Ex. 2: *(X, is founder of, Y)* and *(X, is CEO of, Y)*.

These two relations are not the same, but they are more similar than the case when compared to (X, wrote, Y). Identifying them as similar could be useful as an initial guess for textual entailment [Dagan *et al.*, 2013]. For example, if a text contains "Larry Page founded Google on September 4, 1998," the following hypothesis is likely to be true: (Larry Page, is CEO of, Google).

Ex. 3: *(X, written by, Y)* and *(X, part of, Z)$\wedge$(Y, wrote, Z)*.

This example contains a multi-hop relation, which is a conjunction of multiple relations. If we can find many entities to instantiate such relations, we can group them together. When we retrieve relations using entity pairs, we can interpret these relations interchangeably. For example, we can use "Harry Potter and the Philosopher's Stone" and "J. K. Rowling" to search the knowledge base and check the possible relation expressions between them. We can then interpret (X, written by, Y) as (X, part of, Z)$\wedge$(Y, wrote, Z), and the latter has more information (e.g., we have Z = "Harry Potter Literary Series") about the relation between X and Y. In addition, identifying the multi-hop relations allows hypothesizing possible rules for knowledge inference [Richardson and Domingos, 2006].

All of the above examples boil down to a fundamental relation clustering problem. Clustering can help us identify a lot of such useful semantically similar relation expressions. There have been several relation clustering algorithms proposed, e.g., using one dimensional clustering (e.g., Kmeans) [Bollegala *et al.*, 2009], co-clustering [Dhillon *et al.*, 2003; Bollegala *et al.*, 2010], non-parametric Bayeisan modeling [Kemp *et al.*, 2006], multi-relational clustering using Markov logic network [Kok and Domingos, 2007; 2008], and

[1] https://www.freebase.com/

[2] We use *relation expression* to represent the surface pattern of the *relation*. Sometimes they are of the same meaning.

tensor decomposition based clustering [Sutskever *et al.*, 2009; Kang *et al.*, 2012]. However, there is a major problem in the previous work.

Previous approaches only considered clustering relation expressions based on the intersection of the associated entity sets. However, there exists important background knowledge that can be used to improve clustering. For example, in both relations (X, is founder of, Y) and (X, is CEO of, Y), the left entity X should be a person and the right entity Y should be an organization. If we can constrain the entity types, then some illegitimate relations for a relation cluster can be filtered out.

In this paper, we propose a Constrained Tripartite Graph Clustering (CTGC) algorithm to tackle this problem. We introduce side information via must-link and cannot-link constraints to improve the clustering results. Then the type information about the entities can serve as an indirect supervision for relation clustering. To verify the indirect supervision, we derive the constraints either from ground-truth of entities and relations, or based on knowledge automatically induced from the data. We use two real world datasets to evaluate the clustering results. The first dataset is based on a human annotated knowledge base, Freebase. We generate constraints based on the ground-truth types of entities and relations. This dataset is used to demonstrate the effectiveness of the algorithm. The second dataset is the data extracted by an open IE system, ReVerb[3]. For this data, we generate entity constraints based on the results from a state-of-the-art named entity recognizer [Ratinov and Roth, 2009], and the relation constraints based on the similarity between relation expressions. This dataset shows that the indirect supervision can be automatically obtained. Even the constraints are not perfect, the information can be used to improve relation clustering results. Our contributions can be summarized as twofold:

- We formulate the relation clustering problem as a constrained tripartite graph clustering problem and develop an alternating optimization algorithm to find the clusters of relations.
- We use two datasets to demonstrate our approach: a dataset with Freebase relations and a dataset with open IE relations. The two datasets both show the effectiveness of the clustering algorithm and the usability of real applications.

Related Work

In this section, we discuss the related work from both problem and algorithm perspectives.

Relation Clustering Problems

There has been a lot of work on relation extraction from text, most of which are supervised or semi-supervised methods relying on training data [Mintz *et al.*, 2009; Chan and Roth, 2010; 2011; Li *et al.*, 2011; Li and Ji, 2014]. Researchers also considered using clustering to perform unsupervised relation extraction [Hasegawa *et al.*, 2004; Shinyama and Sekine, 2006; Kok and Domingos, 2008; Yao *et al.*, 2011; Wang *et al.*, 2013]. Some of the relation extraction algorithms tried to find clusters among relation expressions between restricted types of named entities to discover unrestricted types of relations [Hasegawa *et al.*, 2004; Shinyama and Sekine, 2006; Riedel *et al.*, 2013; Rocktäschel *et al.*, 2015]. This is similar to our approach when ours is applied to open information extraction [Banko *et al.*, 2007; 2008; Fader *et al.*, 2011]. Nonetheless, there are two major differences. First, they only considered relation types between fixed types of named entities. However, most of the open domain relations are not restricted to named entities [Banko *et al.*, 2007]. Thus, our method is more flexible and extensible because we cluster the relations from open information extraction directly based on the data statistics and only use named entities as constraints. Second, besides relation extraction, our algorithm can also be applied to knowledge bases to canonicalize different relations with clusters [Galárraga *et al.*, 2014], especially with multi-hop relations (shown in Ex. 3 in the introduction). Therefore, we are trying to solve a more general problem.

Relation Clustering Algorithms

As we mentioned in the introduction, there have been several different formulations of the relation clustering problem resulting in different algorithms. We solve the problem by modeling the data as a tripartite graph clustering problem, which incorporates more information than one-dimensional clustering and co-clustering, and uses more condensed information than tensor based clustering. Moreover, we incorporate constraints as side information into the tripartite graph clustering problem. Such side information is in the forms of must-links and cannot-links, which has been established and proven to be effective in semi-supervised clustering [Basu *et al.*, 2008]. Constraints have been applied to one-dimensional clustering [Basu *et al.*, 2004; Lu and Leen, 2007] and co-clustering [Shi *et al.*, 2010; Song *et al.*, 2013; Chang *et al.*, 2014] and tensor based clustering [Sutskever *et al.*, 2009; Kang *et al.*, 2012], but haven't been explored for tripartite graph clustering problem. More interestingly, we explore how to automatically generate the constraints instead of using the human annotated knowledge.

Constrained Relation Clustering

In this section, we present our problem formulation and solution to constrained relation clustering.

Problem Formulation

Each relation triplet is represented as (e^1, r, e^2). Let the relation set be $\mathcal{R} = \{r_1, r_2, \ldots, r_M\}$, where M is the size of $\mathcal{R}$, and the entity set be $\mathcal{E}^I = \{e^I_1, e^I_2, \ldots, e^I_{V_I}\}$, where V_I is the size of $\mathcal{E}^I$. $\mathcal{E}^1, w.r.t.\ I = 1$, represents the left entity set where $e^1 \in \mathcal{E}^1$. $\mathcal{E}^2, w.r.t.\ I = 2$, represents the right entity set where $e^2 \in \mathcal{E}^2$. We also denote three latent label sets $\mathcal{L}_r = \{l_{r_1}, l_{r_2}, \ldots, l_{r_M}\}$, and $\mathcal{L}_{e^I} = \{l_{e^I_1}, l_{e^I_2}, \ldots, l_{e^I_{V I}}\}$ to indicate the clusters for relations and entities (two sets, $I \in \{1, 2\}$), respectively.

Fig. 1 shows an example of constrained relation clustering problem. The tripartite graph models the correlation among the left entity set $\mathcal{E}^1$, the relation set $\mathcal{R}$, and the right entity set $\mathcal{E}^2$. In the figure, we illustrate four relation triplets: (Larry

[3] http://reverb.cs.washington.edu/

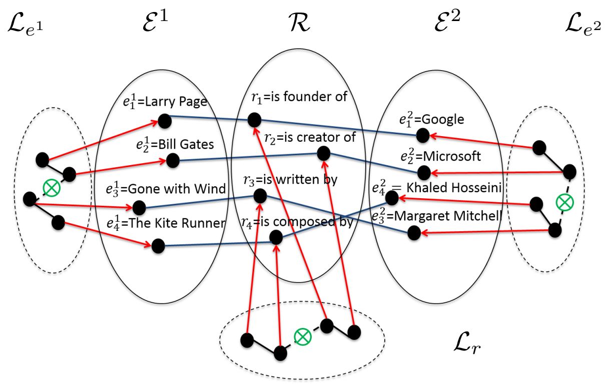


Figure 1: Illustration of the CTGC model. $\mathcal{R}$: relation set; $\mathcal{E}^1$: left entity set, a left entity $e_i^1 \in \mathcal{E}^1$; $\mathcal{E}^2$: right entity set, a right entity $e_j^2 \in \mathcal{E}^2$; $\mathcal{L}_r$: relation latent label set; $\mathcal{L}_{e^1}$: left entity latent label set; $\mathcal{L}_{e^2}$: right entity latent label set.

Page, is founder of, Google), (Bill Gates, is creator of, Microsoft), (Gone with Wind, is written by, Margaret Mitchell), and (The Kite Runner, is composed by, Khaled Hosseini). For (Larry Page, is founder of, Google), e_1^1 = Larry Page, r_1 = is founder of, and e_1^2 = Google, the corresponding latent labels are $l_{e_1^1} = \textit{Person} \in \mathcal{L}_{e^1}$, $l_{r_1} = \textit{Leadership of} \in \mathcal{L}_r$, and $l_{e_1^2} = \textit{Organization} \in \mathcal{L}_{e^2}$, respectively. Then we build a must-link between "is founder of" and "is creator of" if we know they should belong to the same cluster (*Leadership of*), and build a cannot-link between "is founder of" and "is composed by" if we know they are different. Besides, we build a must-link for entities "Larry Page" and "Bill Gates" since the types are the same (*Person*), while we build a cannot-link for "Microsoft" and "Margaret Mitchell" since they have different types (*Organization* and *Person*). We prefer to impose soft constraints to the above relations and entities, since in practice, some constraints could be violated [Chang *et al.*, 2012].

To formulate CTGC, we assume that the triplet joint probability can be decomposed as $p(e_i^1, r_m, e_j^2) \propto p(r_m, e_i^1)p(r_m, e_j^2)$, where the joint probability of $p(r_m, e_i^I)$ can be calculated based on the co-occurrence counts of r_m and e_i^I. We follow Information-Theoretic Co-Clustering (ITCC) [Dhillon *et al.*, 2003] and use

$$q(r_m, e_i^I) = p(\hat{r}_{k_r}, \hat{e^I}_{k_{e^I}})p(r_m|\hat{r}_{k_r})p(e_i^I|\hat{e^I}_{k_{e^I}}), \tag{1}$$

to approximate $p(r_m, e_i^I)$ for the clustering problem. In Eq. (1) , $\hat{r}_{k_r}$ and $\hat{e^I}_{k_{e^I}}$ are cluster indicators, k_r and k_{e^I} are cluster indices.

ITCC minimizes the Kullback-Leibler (KL) divergence $D_{KL}(p(\mathcal{R}, \mathcal{E}^\mathcal{I})||q(\mathcal{R}, \mathcal{E}^\mathcal{I}))$ to evaluate whether the co-clustering produces a good result, where $p(\mathcal{R}, \mathcal{E}^\mathcal{I})$ and $q(\mathcal{R}, \mathcal{E}^\mathcal{I})$ are multinomial distributions composed by $p(r_m, e_i^I)$ and $q(r_m, e_i^I)$ respectively. Minimizing the KL divergence means the approximate function should be as similar as the original probabilities for co-occurrence between entities and relations. In our problem, we use a combination of two terms to evaluate our tripartite graph clustering:

$$D_{KL}(p(\mathcal{R}, \mathcal{E}^1)||q(\mathcal{R}, \mathcal{E}^1)) + D_{KL}(p(\mathcal{R}, \mathcal{E}^2)||q(\mathcal{R}, \mathcal{E}^2)). \tag{2}$$

Since the relation indictor $\hat{r}_{k_r}$ in $q(r_m, e_i^I)$ will be optimized based on the combination of two terms, it will be affected by both left- and right-side entity clusters.

To incorporate the constraints, we design three sets of cost functions for $\mathcal{L}_r$, $\mathcal{L}_{e^1}$, and $\mathcal{L}_{e^2}$. We take relation labels $\mathcal{L}_r$ as an example, and entity labels ($\mathcal{L}_{e^1}$ and $\mathcal{L}_{e^2}$) are similarly defined. For a label l_{r_m}, we denote the must-link set as $\mathcal{M}_{r_m}$, and the cannot-link set as $\mathcal{C}_{r_m}$. For must-links, the cost function is defined as

$$\begin{aligned} & V(r_{m_1}, r_{m_2} \in \mathcal{M}_{r_{m_1}}) \\ = \quad & a_{m_1,m_2} D_{KL}(p(\mathcal{E}^I|r_{m_1})||p(\mathcal{E}^I|r_{m_2})) \cdot \mathcal{I}_{l_{r_{m_1}} \neq l_{r_{m_2}}}, \end{aligned} \tag{3}$$

where $p(\mathcal{E}^I|r_{m_1})$ denotes a multinomial distribution based on the probabilities $(p(e_1^I|r_{m_1}), \ldots, p(e_{V_I}^I|r_{m_1}))^T$, and $\mathcal{I}_{true} = 1$, $\mathcal{I}_{false} = 0$. The above must-link cost function means that if the label of r_{m_1} is not equal to the label of r_{m_2}, then we should take into account the cost function of how dissimilar the two relations r_{m_1} and r_{m_2} are. The dissimilarity is computed based on the probability of entities $\mathcal{E}^I$ given the relations r_{m_1} and r_{m_2} as Eq. (3). The more dissimilar the two relations are, the larger cost is imposed.

For cannot-links, the cost function is defined as

$$\begin{aligned} & V(r_{m_1}, r_{m_2} \in \mathcal{C}_{r_{m_1}}) \\ = \quad & \bar{a}_{m_1,m_2}(D_{max} - D_{KL}(p(\mathcal{E}^I|r_{m_1})||p(\mathcal{E}^I|r_{m_2}))) \cdot \mathcal{I}_{l_{r_{m_1}} = l_{r_{m_2}}}, \end{aligned} \tag{4}$$

where D_{max} is the maximum value for all the $D_{KL}(p(\mathcal{E}^I|r_{m_1})||p(\mathcal{E}^I|r_{m_2}))$. The cannot-link cost function means that if the label of r_{m_1} is equal to the label of r_{m_2}, then we should take into account the cost function of how similar they are. Moreover, a_{m_1,m_2} and $\bar{a}_{m_1,m_2}$ are the tradeoff parameters.

Therefore, both must-links and cannot-links are soft constraints, which are related to the similarity between the relations themselves. If the constraints are violated, then additional costs are added to the final objective function.

Integrating all the constraints for $\mathcal{L}_r$, $\mathcal{L}_{e^1}$ and $\mathcal{L}_{e^2}$ to Eq. (2), the objective function of CTGC is:

$$\begin{aligned} & \{\mathcal{L}_{e^1}, \mathcal{L}_r, \mathcal{L}_{e^2}\} = \arg\min \\ & D_{KL}\left(p(\mathcal{R}, \mathcal{E}^1)||q(\mathcal{R}, \mathcal{E}^1)\right) + D_{KL}\left(p(\mathcal{R}, \mathcal{E}^2)||q(\mathcal{R}, \mathcal{E}^2)\right) \\ & + \textstyle\sum_{r_{m_1}=1}^{M} \sum_{r_{m_2} \in \mathcal{M}_{r_{m_1}}} V(r_{m_1}, r_{m_2} \in \mathcal{M}_{r_{m_1}}) \\ & + \textstyle\sum_{r_{m_1}=1}^{M} \sum_{r_{m_2} \in \mathcal{C}_{r_{m_1}}} V(r_{m_1}, r_{m_2} \in \mathcal{C}_{r_{m_1}}) \\ & + \textstyle\sum_{e_{i_1}^1=1}^{V_1} \sum_{e_{i_2}^1 \in \mathcal{M}_{e_{i_1}^1}} V(e_{i_1}^1, e_{i_2}^1 \in \mathcal{M}_{e_{i_1}^1}) \\ & + \textstyle\sum_{e_{i_1}^1=1}^{V_1} \sum_{e_{i_2}^1 \in \mathcal{C}_{e_{i_1}^1}} V(e_{i_1}^1, e_{i_2}^1 \in \mathcal{C}_{e_{i_1}^1}) \\ & + \textstyle\sum_{e_{j_1}^2=1}^{V_2} \sum_{e_{j_2}^2 \in \mathcal{M}_{e_{j_1}^2}} V(e_{j_1}^2, e_{j_2}^2 \in \mathcal{M}_{e_{j_1}^2}) \\ & + \textstyle\sum_{e_{j_1}^2=1}^{V_2} \sum_{e_{j_2}^2 \in \mathcal{C}_{e_{j_1}^2}} V(e_{j_1}^2, e_{j_2}^2 \in \mathcal{C}_{e_{j_1}^2}) \end{aligned} \tag{5}$$

where $\mathcal{M}_{e_{i_1}^1}$ and $\mathcal{C}_{e_{i_1}^1}$ are the must-link and cannot-link sets for entity $e_{i_1}^1$ labeled with $l_{e_{i_1}^1}$. Similarly, the label of entity $e_{j_1}^2$ also has must-link and cannot-link sets, which are denoted as $\mathcal{M}_{e_{j_1}^2}$ and $\mathcal{C}_{e_{j_1}^2}$, respectively.

Alternating Optimization

Since globally optimizing the latent labels as well as the approximating function $q(r_m, e_i^I)$ is intractable, we perform an alternating optimization shown in Algorithm 1. For each set of labels, we first update the cluster labels based on the fixed

Algorithm 1 Alternating Optimization for CTGC.

```
Input: Tripartite graph defined on relations $\mathcal{R}$, left entities $\mathcal{E}^1$ and right entities $\mathcal{E}^2$; Set maxIter and max$\delta$.
while $t <$ maxIter and $\delta >$ max$\delta$ do
    $\mathcal{R}$ Label Update: minimize Eq. (5) w.r.t. $\mathcal{L}_r$.
    $\mathcal{R}$ Model Update: update parameters in Eqs. (7-9).

    $\mathcal{E}^1$ Label Update: minimize Eq. (5) w.r.t. $\mathcal{L}_{e^1}$.
    $\mathcal{E}^1$ Model Update: update parameters in Eqs. (7-9).

    $\mathcal{R}$ Label Update: minimize Eq. (5) w.r.t. $\mathcal{L}_r$.
    $\mathcal{R}$ Model Update: update parameters in Eqs. (7-9).

    $\mathcal{E}^2$ Label Update: minimize Eq. (5) w.r.t. $\mathcal{L}_{e^2}$.
    $\mathcal{E}^2$ Model Update: update parameters in Eqs. (7-9).
    Compute cost change $\delta$ using Eq. (5).
end while
```

model function $q(r_m, e_i^I)$. Taking the optimizing $\mathcal{L}_r$ as an example, we use the iterated conditional mode (ICM) algorithm [Basu *et al.*, 2004] to find the cluster labels. We update one label l_{r_m} at a time, and keep all the other labels fixed:

$$
\begin{aligned}
l_{r_m} = \arg \min_{l_{r_m}=k_r} & D_{KL}(p(\mathcal{E}^1|r_m)||p(\mathcal{E}^1|\hat{r}_{k_r})) \\
+ & D_{KL}(p(\mathcal{E}^2|r_m)||p(\mathcal{E}^2|\hat{r}_{k_r})) \\
+ & \sum_{\substack{r_{m'} \in \mathcal{M}_{r_m}; \\ I_{l_{r_m} \neq l_{r_{m'}}}}} a_{m,m'} D_{KL}(p(\mathcal{E}^1|r_m)||p(\mathcal{E}^1|r_{m'})) \\
+ & \sum_{\substack{r_{m'} \in \mathcal{M}_{r_m}; \\ I_{l_{r_m} \neq l_{r_{m'}}}}} a_{m,m'} D_{KL}(p(\mathcal{E}^2|r_m)||p(\mathcal{E}^2|r_{m'})) \\
+ & \sum_{\substack{r_{m'} \in \mathcal{C}_{r_m}; \\ I_{l_{r_m} = l_{r_{m'}}}}} \bar{a}_{m,m'} \left(D^1_{max} - D_{KL}(p(\mathcal{E}^1|r_m)||p(\mathcal{E}^1|r_{m'}))\right) \\
+ & \sum_{\substack{r_{m'} \in \mathcal{C}_{r_m}; \\ I_{l_{r_m} = l_{r_{m'}}}}} \bar{a}_{m,m'} \left(D^2_{max} - D_{KL}(p(\mathcal{E}^2|r_m)||p(\mathcal{E}^2|r_{m'}))\right).
\end{aligned}
\tag{6}
$$

where the information of $q(r_m, e_i^I)$ is incorporated into KL divergences $D_{KL}(p(\mathcal{E}^1|r_m)||p(\mathcal{E}^1|\hat{r}_{k_r}))$ and $D_{KL}(p(\mathcal{E}^2|r_m)||p(\mathcal{E}^2|\hat{r}_{k_r}))$. To understand why $D_{KL}\left(p(\mathcal{R}, \mathcal{E}^1)||q(\mathcal{R}, \mathcal{E}^1)\right) + D_{KL}\left(p(\mathcal{R}, \mathcal{E}^2)||q(\mathcal{R}, \mathcal{E}^2)\right)$ can be re-written as $D_{KL}(p(\mathcal{E}^1|r_m)||p(\mathcal{E}^1|\hat{r}_{k_r})) + D_{KL}(p(\mathcal{E}^2|r_m)||p(\mathcal{E}^2|\hat{r}_{k_r}))$, please refer to ITCC for more details [Dhillon *et al.*, 2003].

Then, with the labels $\mathcal{L}_r$ and $\mathcal{L}_{e^I}$ fixed, we update the model function $q(r_m, e_i^I)$. The update of q is not influenced by the must-links and cannot-links. Thus we can modify them the same as ITCC [Dhillon *et al.*, 2003]:

$$q(\hat{r}_{k_r}, \hat{e^I}_{k_{e^I}}) = \sum_{l_{r_m}=k_r} \sum_{l_{e_i^I}=k_{e^I}} p(r_m, e_i^I) \tag{7}$$

$$q(r_m|\hat{r}_{k_r}) = \frac{q(r_m)}{q(l_{r_m} = k_r)} \;\; [q(r_m|\hat{r}_{k_r}) = 0 \text{ if } l_{r_m} \neq k_r] \tag{8}$$

$$q(e_i^I|\hat{e^I}_{k_{e^I}}) = \frac{q(e_i^I)}{q(l_{e_i^I} = k_{e^I})} \;\; [q(e_i^I|\hat{e^I}_{k_{e^I}}) = 0 \text{ if } l_{e_i^I} \neq k_{e^I}] \tag{9}$$

where $q(r_m) = \sum_{e_i^I} p(r_m, e_i^I)$, $q(e_i^I) = \sum_{r_m} p(r_m, e_i^I)$, $q(\hat{r}_{k_r}) = \sum_{k_{e^I}} p(\hat{r}_{k_r}, \hat{e^I}_{k_{e^I}})$ and $q(\hat{e^I}_{k_{e^I}}) = \sum_{k_r} p(\hat{r}_{k_r}, \hat{e^I}_{k_{e^I}})$.

Algorithm 1 summarizes the main steps in the procedure. The objective function (5) with our alternating update monotonically decreases to a local optimum. This is because the ICM algorithm decreases the non-negative objective function (5) to a local optimum given a fixed q function. Then the update of q is monotonically decreasing as guaranteed by the theorem proven in [Song *et al.*, 2013].

The time complexity of Algorithm 1 is $O((n_{nz} + (n_c * iter_{ICM})) \cdot (K_{e^1} + K_{e^2} + K_r)) \cdot iter_{AEM}$, where n_{nz} is the total number of non-zero elements in the entity-relation co-occurrence matrix, n_c is the constraint number, $iter_{ICM}$ is the ICM iteration number in the E-Step, K_{e^1}, K_{e^2} and K_r are the cluster numbers, and $iter_{AEM}$ is the iteration number of the alternating optimization algorithm.

Experiments

In this section, we evaluate the proposed approach on two real world datasets.

Rel-KB and Constraints

Freebase is a publicly available knowledge base containing over 2 billions relation expressions between 40 millions entities. The Rel-KB dataset is constructed as follows: we select six popular one-hop relation categories in Freebase, i.e., *Organization-Founder*, *Book-Author*, *Actor-Film*, *Location-Contains*, *Music-Track*, *Person-Profession*. Then, for each relation category, we randomly sample $5,000$ entity pairs. For each entity pair in the selected data, we enumerate all the $l = L/2$-hop relations for each entity, and combine them to generate the multi-hop relations within length-L (experimentally, $L = 4$). Finally, we have $16,516$ relation expressions with relation categories.[4]

Then, we derive relation and entity constraints from the Rel-KB dataset. Based on Freebase, it is straightforward to design constraints for both relations and entities.

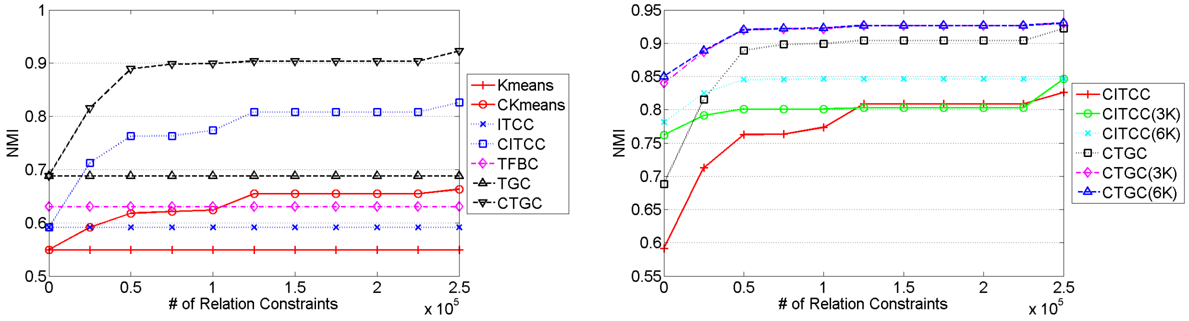
Relation constraints. **(1) Must-links.** If two relations are generated from the same relation category, we add a must-link. For example, (X, 's founder is, Z) ∧ (Z, is influence peer of, Y) and (X, 's founder is, Y) are generated from entity pair (Google, Larry Page) and (Microsoft, Bill Gates). Both entity pairs belong to *Organization-Founder* relation category. Thus a must-link can be added to the two relations. **(2) Cannot-links.** If two relations are generated from entity pairs with different categories, we add a cannot-link to them.

Entity constraints. **(1) Must-links.** If two entities belong to the same entity category, we add a must-link. **(2) Cannot-links.** If two entities belong to different entity categories, we add a cannot-link. For example, the entity categories of "Google" and "My Worlds" are *Organization* and *Music* respectively. In this case, we add a cannot-link to them.

Analysis of Clustering Results on Rel-KB

Here, we present the results on the Rel-KB dataset which has gold standard for cluster labels of relation expressions. This demonstrates the performance of our algorithm in the ideal situation, since our constraints are derived based on the gold standard provided by Freebase. We employ the widely-used normalized mutual information (NMI) [Strehl and Ghosh, 2003] as the measure. The NMI score is 1 if the clustering results match the category labels perfectly and 0 if the clusters are obtained from a random partition. In general, the larger the scores are, the better the clustering results are.

[4] We assume all the relations generated with a certain entity pair being within the same category as the gold standard.



(a) Effects of relation constraints.

(b) Effects of entity constraints.

Figure 2: Comparison of relation clustering algorithms (six relation categories). Methods: Kmeans and constrained Kmeans (CKmeans), Information-Theoretic Co-Clustering (ITCC) and Constrained ITCC (CITCC), Tensor Factorization based Clustering (TFBC). Our method CTGC outperforms the other methods.

We call our algorithm Constrained Tripartite Graph Clustering (CTGC), and also denote the unconstrained version as TGC. In this experiment, we compare the performance of CTGC with that of several representative approaches such as (1) *one-dimensional clustering* algorithms Kmeans and constrained Kmeans (CKmeans) [Basu *et al.*, 2004], (2) *co-clustering* algorithms ITCC [Dhillon *et al.*, 2003] and Constrained ITCC (CITCC) [Song *et al.*, 2013], and (3) *multi-relational clustering* algorithm which is Tensor Factorization based Clustering (TFBC)[5] [Sutskever *et al.*, 2009]. In Kmeans, each relation expression is represented as an entity frequency vector. In co-clustering, the relation-entity co-occurrence (only the left entity is used) is used as the evidence of co-clustering. In three-dimensional tensor, an element in the tensor simply represents a triplet (e^1, r, e^2) appearing in the data. In our algorithm, we treat the data as a tripartite graph, which condenses the tensor into two slices of matrices. Similar to co-clustering, each relation expression can be associated with multiple entities on both left and right sides. For relation constraints, we generate both must-links and cannot-links based on the method described before. In the following experiments, both the relation and entity cluster numbers are set to 6 (for both $\mathcal{E}^1$ and $\mathcal{E}^2$), the ground-truth number, respectively. Moreover, the trade-off parameters a_{m_1,m_2} and $\bar{a}_{m_1,m_2}$ for constraints in Eqs. (3) and (4) are empirically set to $1/\sqrt{M}$ for relations and $1/\sqrt{V_I}$ $(I \in \{1, 2\})$ for entities following [Song *et al.*, 2013].

To compare the relation clustering results, we vary the number of relation and entity constraints by randomly selecting a fixed number of constraints from all possible must-links and cannot-links to investigate their impacts on clustering performance. Fig. 2 shows the experimental results. The x-axis in each sub-figure represents the number of relation constraints used in each experiment and the y-axis represents the averaged NMI value of five random trials.

As shown in Fig. 2(a), among all the methods we test, CTGC consistently performs the best. When there is no constraint, we can see that ITCC is better than Kmeans. The reason is that the co-clustering algorithm ITCC considers the information which also takes entity clusters into account. Moreover, TFBC is better than ITCC since it considers both left and right sides of entities clusters while ITCC only considers one side of entities. Furthermore, our TGC is better than TFBC. This is because we condense the tensor into two slices of matrices which represent the tripartite graph. Tensor may generate more wrong cluster assignments when the data is too sparse. The CITCC outperforms the TGC method because TGC does not use any constraints. In Fig. 2(b), we can see that CTGC significantly outperforms CITCC when we start adding more constraints to TGC. In addition, we can see that relation constraints can improve the clustering performance. In general, the more relation constraints we add, the better the clustering results are.

Fig. 2(b) shows the effect of entity constraints along with the relation constraints. Besides the relation constraints which are the same as the ones in Fig. 2(a), we also add $3,000$ (i.e., 3K) and $6,000$ (i.e., 6K) entity constraints for CITCC and CTGC respectively. We can see that entity constraints are also very helpful for improving the relation clustering performance. The reason is that entity clustering information is transferred through the co-occurrence of entities and relations to the relation side. In general, with more entity constraints, the clustering results are better. Particularly, we see that when there is no relation constraint, we can even boost the NMI score from 0.69 to 0.85 with CTGC using only entity constraints. Therefore, it shows that even if we have little knowledge about relations, we can still expect better results if we know some knowledge about entities.

By looking into the clustering results, interestingly, in the *Music-Track* cluster, CTGC could find the four-hop relation: (X, made^{-1}, Person) $\wedge$ (Person, same height, Person) $\wedge$ (Person, perform in, Video) $\wedge$ (Video, play in TV^{-1}, Y), which means the person who makes the music has the same height with the person who performs in the music video of the track.[6] It is semantically similar to the *Music-Track* cluster but we believe there should be very few (only one in our data) entities which could instantiate this relation. Therefore, it is very difficult to cluster this relation with the others. However, by introducing the constraints, we know the entities instantiating

[5] We use the standard Tensor Toolbox for Matlab: http://www.sandia.gov/~tgkolda/TensorToolbox/.

[6] We use entity types instead of entities in the intermediate relations of a multi-hop relation to be more easily understood.

(a) Examples generated by TGC.

Organization-Founder	(X, founded^{-1}, Y); (X, was founded by^{-1}, Y); (X, directed by, Y); (X, , led by, Y); (X, is established by, Y); (X, left^{-1}, Y).
Book-Author	(X, wrote^{-1}, Y); (X, is a play by, Y); (X, is a book by, Y); (X, renamed to^{-1}, Y); (X, is a poem by, Y); (X, born in^{-1}, Y).
Actor-Film	(X, star^{-1}, Y); (X, feature^{-1}, Y); (X, stars^{-1}, Y); (X, who played, Y); (X, starred in, Y); (X, 's capital in, Y).
Location-Contains	(X, locate capital in, Y); (X, build ,Y); (X, is contained by^{-1}, Y); (X, have, Y); (X, extend,Y); (X, competed for, Y).
Music-Track	(X, released, Y); (X, containing, Y); (X, has a song, Y); (X, from^{-1}, Y); (X, is popular in^{-1}, Y); (X, painting, Y).
Person-Profession	(X, is good at, Y); (X, referred to, Y); (X, major in, Y); (X, is a celebrity, Y); (X, is talent in, Y); (X, perform in, Y).

(b) Examples generated by CTGC.

Organization-Founder	(X, founded by, Y); (X, led by, Y); (X, is the owner of^{-1}, Y); (X, , sold by, Y); (X, , owned by, Y); (X, who left^{-1}, Y).
Book-Author	(X, is the author of^{-1}, Y); (X, written by, Y); (X, edited by, Y); (X, composed by, Y); (X, is a fantasy novel by, Y); (X, writes^{-1}, Y); (X, composed^{-1}, Y); (X, , who wrote^{-1}, Y); (X, is a book written by, Y); (X, was developed by, Y).
Actor-Film	(X, , which stars^{-1}, Y); (X, act in, Y); (X, makes a brief appearance, Y); (X, , appears in, Y); (X, performed by^{-1}, Y); (X, won best actor for, Y); (X, , who played in, Y); (X, a movie starring^{-1}, Y); (X, performed the title role in, Y).
Location-Contains	(X, locate capital in, Y); (X, 's capital in, Y); (X, is a department of^{-1}, Y); (X, is a state of^{-1},Y); (X, 's downtown, Y).
Music-Track	(X, released, Y); (X, containing, Y); (X, was released in^{-1},Y); (X, is recorded in^{-1},Y); (X, , a record in^{-1},Y); (X, is a single of^{-1}, Y); (X, is a hit of^{-1}, Y); (X, is a produce in^{-1}, Y); (X, hit, Y); (X, a written work recorded in^{-1}, Y).
Person-Profession	(X, legend^{-1}, Y); (X, retires from, Y); (X, 's profession is, Y); (X, is famous in, Y); (X, win champion, Y); (X, play, Y).

Table 1: Examples of relation clusters from Rel-OIE. We use "-1" to represent the inverse order of the relation. Notice that, we have all the cases generated by the other five clustering algorithms. Due to the space limitation, we only show the results of TGC and CTGC.

this relation are must-linked to other entities which have relation expressions in the *Music-Track* cluster. This relation is finally clustered in the *Music-Track* cluster.

Rel-OIE and Constraints

In practice, the relations, entities, and constraints derived from knowledge base are still limited. Therefore, we also develop another dataset called Rel-OIE in a more realistic scenario. We employ the open IE system, Reverb [Fader *et al.*, 2011], to generate relation triplets from Wikipedia sentences containing at least one entity in Rel-KB.[7] We do this because Wikipedia text is cleaner compared to generic Web documents, and the sentences containing knowledge base entities may have higher possibility to have the relations of interests. In Rel-OIE, we have 137,674 unique relation expressions, 267,133 left entities, and 229,979 right entities.

Since in the open information extraction setting, we only have the sentences in free text format, we construct the constraints using the following methods.

Relation Must-links. If the similarity between two relation phrases is beyond a predefined threshold (experimentally, 0.5), we add a must-link to these relations. The similarity here is defined as the token-based Jaccard similarity between two phrases. For example, two phrases "'s founder is" and "'s founder is influence peer of" share three common tokens and thus they may both imply the same relation cluster. In this case, we add a must-link between these two phrases.

Entity Must-links. If two entities are of the same named entity type, we add a must-link to these entities. We use one of the state-of-the-art named entity recognizers [Ratinov and Roth, 2009], since it provides a larger number of types of named entities (18 types trained based on Ontonotes).

Case Study of Clustering Results on Rel-OIE

We herein present some examples from Rel-OIE dataset. We also cluster the relations into six clusters since we only extract the relations from sentences containing the entities in the Rel-KB dataset. In this case, it is easier for us to understand what happened after clustering.

We show the clustering results of TGC in Table 1(a) and the results of CTGC in Table 1(b). In general, the clustering results of TGC and CTGC both make sense. For example, in the *Location-Contains* cluster, CTGC as well as TGC find similar relations, e.g., (X, locate capital in, Y), (X, is a state of^{-1}, Y).

The clustering results of CTGC seem much better. For example, TGC does not cluster (X, locate capital in, Y) and (X, 's capital in, Y) together while CTGC does. By further checking the data, we found the reasons for the better results are: (1) there are relation must-links between (X, locate capital in, Y) and the other relation expressions such as (X, 's capital in, Y) in the cluster; (2) there are must-links between entities which can instantiate the expressions, e.g., a must-link between "America" and "China" (both are *Geographical/Social/Political Entities (GPE)*), and a must-link between "New York" and "Beijing" (both are *GPE*). This proves the importance of constraints for improving the clustering performance in the noisy scenario.

Conclusion

In this paper, we study the relation clustering problem. We model the relation clustering as a constrained tripartite graph clustering problem, and propose to use side information to help improve the clustering performance. We use two datasets to demonstrate the effectiveness of our approach, and show that this direction is promising.

Acknowledgments

Chenguang Wang gratefully acknowledges the support by the National Natural Science Foundation of China (NSFC Grant Number 61472006) and the National Basic Research Program (973 Program No. 2014CB340405). The research is also partially supported by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053, and by DARPA

[7] We do not use Ollie [Schmitz *et al.*, 2012] because Reverb is faster and with acceptable precision in our data.

under agreement number FA8750-13-2- 0008. The views and conclusions are those of the authors and do not necessarily reflect the view of the agencies.

References

[Banko *et al.*, 2007] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction for the web. In *IJCAI*, pages 2670–2676, 2007.

[Banko *et al.*, 2008] Michele Banko, Oren Etzioni, and Turing Center. The tradeoffs between open and traditional relation extraction. In *ACL*, pages 28–36, 2008.

[Basu *et al.*, 2004] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, pages 59–68, 2004.

[Basu *et al.*, 2008] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008.

[Bollegala *et al.*, 2009] Danushka T Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Measuring the similarity between implicit semantic relations from the web. In *WWW*, pages 651–660, 2009.

[Bollegala *et al.*, 2010] Danushka Tarupathi Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Relational duality: Unsupervised extraction of semantic relations between entities on the web. In *WWW*, pages 151–160, 2010.

[Chan and Roth, 2010] Yee Seng Chan and Dan Roth. Exploiting background knowledge for relation extraction. In *COLING*, pages 152–160, 2010.

[Chan and Roth, 2011] Yee Seng Chan and Dan Roth. Exploiting syntactico-semantic structures for relation extraction. In *ACL*, pages 551–560, 2011.

[Chang *et al.*, 2012] Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 2012.

[Chang *et al.*, 2014] Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*, pages 1568–1579, 2014.

[Dagan *et al.*, 2013] Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220, 2013.

[Dhillon *et al.*, 2003] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *KDD*, pages 89–98, 2003.

[Fader *et al.*, 2011] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545, 2011.

[Galárraga *et al.*, 2014] Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. Canonicalizing open knowledge bases. In *CIKM*, pages 1679–1688, 2014.

[Hasegawa *et al.*, 2004] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations among named entities from large corpora. In *ACL*, pages 415–422, 2004.

[Kang *et al.*, 2012] U Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *KDD*, pages 316–324, 2012.

[Kemp *et al.*, 2006] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, pages 381–388, 2006.

[Kok and Domingos, 2007] Stanley Kok and Pedro Domingos. Statistical predicate invention. In *ICML*, pages 433–440, 2007.

[Kok and Domingos, 2008] Stanley Kok and Pedro Domingos. Extracting semantic networks from text via relational clustering. In *ECML*, pages 624–639. 2008.

[Li and Ji, 2014] Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *ACL*, pages 402–412, 2014.

[Li *et al.*, 2011] Qi Li, Sam Anzaroot, Wen-Pin Lin, Xiang Li, and Heng Ji. Joint inference for cross-document information extraction. In *CIKM*, pages 2225–2228, 2011.

[Lu and Leen, 2007] Zhengdong Lu and Todd K Leen. Penalized probabilistic clustering. *Neural Computation*, 19(6):1528–1567, 2007.

[Mintz *et al.*, 2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL*, pages 1003–1011, 2009.

[Ratinov and Roth, 2009] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *CoNLL*, pages 147–155, 2009.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[Riedel *et al.*, 2013] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *NAACL*, pages 74–84, 2013.

[Rocktäschel *et al.*, 2015] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *NAACL*, 2015.

[Schmitz *et al.*, 2012] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. Open language learning for information extraction. In *EMNLP*, pages 523–534, 2012.

[Shi *et al.*, 2010] Xiaoxiao Shi, Wei Fan, and Philip S Yu. Efficient semi-supervised spectral co-clustering with constraints. In *ICDM*, pages 1043–1048, 2010.

[Shinyama and Sekine, 2006] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *HLT-NAACL*, pages 304–311, 2006.

[Socher *et al.*, 2013] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, pages 926–934, 2013.

[Song *et al.*, 2013] Yangqiu Song, Shimei Pan, Shixia Liu, Furu Wei, M.X. Zhou, and Weihong Qian. Constrained text coclustering with supervised and unsupervised constraints. *TKDE*, 25(6):1227–1239, June 2013.

[Strehl and Ghosh, 2003] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *JMLR*, 3:583–617, 2003.

[Sutskever *et al.*, 2009] Ilya Sutskever, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Modelling relational data using bayesian clustered tensor factorization. In *NIPS*, pages 1821–1828, 2009.

[Wang *et al.*, 2013] Chenguang Wang, Nan Duan, Ming Zhou, and Ming Zhang. Paraphrasing adaptation for web search ranking. In *ACL*, pages 41–46, 2013.

[West *et al.*, 2014] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *WWW*, pages 515–526, 2014.

[Yao *et al.*, 2011] Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. Structured relation discovery using generative models. In *EMNLP*, pages 1456–1466, 2011.