Information Systems 80 (2019) 64–75

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Knowledge triple mining via multi-task learning

Zhao Zhang [a,b], Fuzhen Zhuang [a,b,*], Xuebing Li [a], Zheng-Yu Niu [c], Jia He [a,b], Qing He [a,b], Hui Xiong [d]

[a] *Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China*
[b] *University of Chinese Academy of Sciences, Beijing 100049, China*
[c] *Baidu Inc., Beijing, China*
[d] *Management Science and Information Systems Department, Rutgers Business School, Rutgers University, Newark, NJ, USA*

HIGHLIGHTS

- We propose S^2 AMT to solve the problem of KTM with limited seed instances.
- Our framework obtain better performance using MTL methods.
- Our framework jointly use labeled and unlabeled instances during the training stage.
- We give a fast method to find related tasks to further improve the performance.
- Our work provides a new perspective for KTM when having limited seed instances.

ARTICLE INFO

Article history:
Received 20 November 2017
Received in revised form 10 July 2018
Accepted 15 September 2018
Available online 19 September 2018

Keywords:
Multi-task learning
Knowledge mining
Relation extraction
Knowledge graph construction

ABSTRACT

Recent years have witnessed the rapid development of knowledge bases (KBs) such as WordNet, Yago and DBpedia, which are useful resources in AI-related applications. However, most of the existing KBs are suffering from incompleteness and manually adding knowledge into KBs is inefficient. Therefore, automatically mining knowledge becomes a critical issue. To this end, in this paper, we propose to develop a model (S^2 AMT) to extract knowledge triples, such as <Barack Obama, wife, Michelle Obama>, from the Internet and add them to KBs to support many downstream applications. Particularly, because the seed instances[1] for every relation is difficult to obtain, our model is capable of mining knowledge triples with limited available seed instances. To be more specific, we treat the knowledge triple mining task for each relation as a single task and use multi-task learning (MTL) algorithms to solve the problem, because MTL algorithms can often get better results than single-task learning (STL) ones with limited training data. Moreover, since finding proper task groups is a fatal problem in MTL which can directly influences the final results, we adopt a clustering algorithm to find proper task groups to further improve the performance. Finally, we conduct extensive experiments on real-world data sets and the experimental results clearly validate the performance of our MTL algorithms against STL ones.

1. Introduction

Nowadays, knowledge bases (KBs) are extremely useful resources for query expansion, coreference resolution, question answering and information retrieval. Several KBs have been proposed in recent years, such as WordNet [1], Yago [2], NELL [3], DBpedia [4] and Google's Knowledge Graph.[2] Knowledge is often represented in the form of knowledge triples, such as <Barack Obama, wife, Michelle Obama>. In this paper, we represent this kind of knowledge triples as $< s, p, o >$, where s and o are two entities and p denotes the relation between s and o. Since manually adding knowledge into bases is a tedious and endless task, mining knowledge automatically becomes an important issue.

Baidu[3] has a large-scale Chinese KB. However, the KB of Baidu is far from complete, especially for some specific relations. In this paper, we focus on 37 relations of which the number of corresponding knowledge triples is relatively small in Baidu's KB, the 37 relations are shown in Table 1, the translation of these relations are shown in the last column. We want to extract $< s, p, o >$ triples from free text in Chinese web pages automatically. The knowledge we

* Correspondence to: NO. 6 Kexueyuan South Road, Haidian District, Beijing, 100190, PR China.
E-mail address: zhuangfuzhen@ict.ac.cn (F. Zhuang).

1 In this paper, seed instances refer to labeled positive instances.

2 https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html.

3 www.baidu.com.

https://doi.org/10.1016/j.is.2018.09.003
0306-4379/© 2018 Elsevier Ltd. All rights reserved.

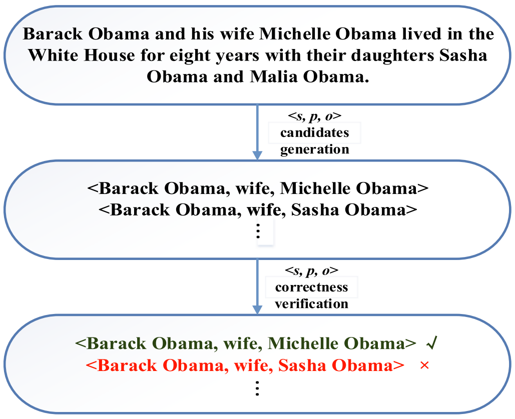
Barack Obama and his wife Michelle Obama lived in the White House for eight years with their daughters Sasha Obama and Malia Obama.
<s, p, o> candidates generation
<Barack Obama, wife, Michelle Obama>
<Barack Obama, wife, Sasha Obama>
⋮
<s, p, o> correctness verification
<Barack Obama, wife, Michelle Obama> ✓
<Barack Obama, wife, Sasha Obama> ×
⋮

Fig. 1. An example of knowledge extraction.

get will be added to the KB of Baidu to support many downstream applications, like the search engine and the question-answering system, etc.

Generally, knowledge triple mining methods consist of two substeps, $< s, p, o >$ candidates generation and $< s, p, o >$ correctness verification, as is shown in Fig. 1. In this paper, we first introduce the details of these two stages and then further give numerical metrics of the $< s, p, o >$ correctness verification stage.

Though many supervised information extraction (IE) methods have been proposed in recent years, most methods have the following weakness that the availability of sufficient seed instances is always assumed before training models, which does not always hold in practice [5]. Our task is to extract $< s, p, o >$ triples of 37 relations from real-world web pages with limited training data. Though a large number of sentences exist on the Internet, we can only get $< s, p, o >$ triples from a tiny part of them. In addition, as we all know, manually annotating instances for each relation is costly and time consuming. Therefore, it is very difficult for us to get plenty of seed instances.

Multi-task learning (MTL) handles multiple related learning tasks simultaneously so that the useful information can be leveraged between each task, and the performance of each individual task can be promoted. Based on previous studies [6–8], MTL algorithms can often get good results with limited seed instances. Since the $< s, p, o >$ mining problem for each relation can be treated as a single task, it is possible to use MTL methods on these tasks. Therefore, the first question arises, can MTL algorithms outperform single-task learning (STL) ones in the problem of $< s, p, o >$ mining with limited seed instances? Furthermore, determining proper task groups is a fatal problem in MTL which can directly influence the results [9,10]. Therefore, the second problem is how to find related tasks to form a proper task group?

Along this line, we first propose a model called Semi-Supervised Autoencoder for regularized Multi-Task learning (S^{2}AMT) to solve the knowledge triple mining problem. Then in order to further improve the results, we use a fast clustering algorithm to find proper task groups and run MTL algorithms on these task groups.

In summary, our work is to address the above two problems, and the contributions are highlighted as follows,

- First, we design a novel framework, which can jointly make full use of labeled and unlabeled instances for knowledge triple mining.
- Second, we propose to use MTL methods to obtain better performance when there are only limited seed instances, which provides a new perspective for knowledge triple mining.
- Third, we develop a fast clustering method to find proper task groups to further improve the performance.

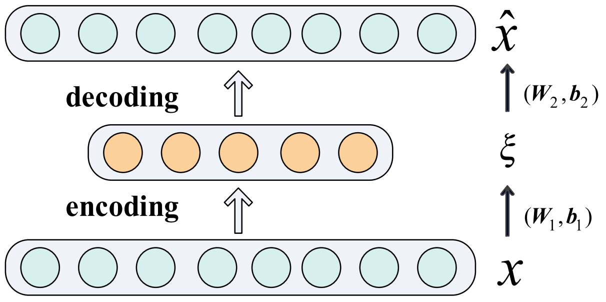

Fig. 2. The framework of Autoencoder.

- Finally, extensive experiments are conducted on real-world data to demonstrate the effectiveness of the proposed model.

The rest of this paper is organized as follows. Section 2 presents the preliminary knowledge, followed by the details of our proposed model in Section 3. Section 4 presents the experimental results, and the related work is given in Section 5. Finally, Section 6 concludes this work.

2. Preliminary knowledge

In this section, we introduce the preliminary knowledge of our work.

2.1. Autoencoder

Autoencoder [11] is a kind of neural network that can find the latent representation of input data. The framework of autoencoder is shown in Fig. 2. An autoencoder comprises an input layer, an output layer and at least one hidden layer. Fig. 2 shows an autoencoder with one hidden layer. An autoencoder can transform the input data into output data with the least possible amount of deviation. Thus the hidden layer can be treated as the latent representation of the input data. The training process of an autoencoder does not need human annotated information, so an autoencoder is an unsupervised learning model. The training process of an autoencoder can be summarized as follows.

As shown in Fig. 2, $\boldsymbol{x}_i \in \mathbb{R}^{m\times 1}$ is the input data, $\boldsymbol{W}_1 \in \mathbb{R}^{k\times m}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{k\times 1}$ are the parameters for encoding, while the corresponding parameters for decoding are $\boldsymbol{W}_2 \in \mathbb{R}^{m\times k}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{m\times 1}$. m is the dimension of the input and output data, and k is the dimension of the hidden layer. The aim of autoencoder is to obtain the latent representation of the input data. The latent representation $\boldsymbol{\xi}$ and the output data $\widehat{\boldsymbol{x}}$ can be computed as Eq.(1), where f is the sigmoid function.

$$\boldsymbol{\xi_i} = f\left(\boldsymbol{W_1 x_i} + \boldsymbol{b_1}\right), \widehat{\boldsymbol{x_i}} = f\left(\boldsymbol{W_2 \xi_i} + \boldsymbol{b_2}\right) \tag{1}$$

The objective of autoencoder forces the output $\boldsymbol{x_i}$ to be as close as possible to the input $\widehat{\boldsymbol{x_i}}$, thus the loss function of autoencoder lies as Eq.(2), where n is the total number of labeled and unlabeled data, $\mathcal{D}$ is the deviation between the input data and output data.

$$\min_{\boldsymbol{W_1},\boldsymbol{b_1},\boldsymbol{W_2},\boldsymbol{b_2}} \mathcal{D} = \sum_{i=1}^{n} \|\widehat{\boldsymbol{x_i}} - \boldsymbol{x_i}\|^2 \tag{2}$$

2.2. Regularized multi-task logistic regression

Logistic regression [12] is a traditional algorithm for binary classification. Given a data point $\boldsymbol{x}_t^{(i)} \in \mathbb{R}^{m\times 1}$, we assume that $y_t^{(i)} \in \{0, 1\}$ is the label of the data point, where t represents task index. The probability that the data point belongs to class 1 is

$$P(y_t^{(i)}|\boldsymbol{x}_t^{(i)}) = f(\boldsymbol{w}_t^\top \boldsymbol{x}_t^{(i)}), \tag{3}$$

Table 1
37 relations.

relation ID	relation (Chinese)	relation (English)
1	表演者	actor/actress
2	门派	martial arts school
3	综艺嘉宾	guests of a variety show
4	替身	body double
5	专业代码	subject ID
6	航天员	astronaut
7	老乡	fellow-townsman
8	亲家	parents of one's daughter/son-in-law
9	花色	color of a flower
10	传承人	inheritor
11	气候	climate
12	爱好	hobby
13	体型	body shape
14	笔名	pseudonym
15	原产地	place of origin
16	身高	body height
17	妹夫	brother-in-law
18	军衔	military rank
19	自称	a name that a person calls himself
20	掌门	head owner of a martial arts school
21	门票价格	ticket price
22	作品	works
23	分布区域	distributed area
24	传播途径	mode of transmission
25	拍摄地	filming location
26	传闻不和	rumored to have a bad relationship with
27	是几级保护动物	wild life conservation status
28	选官制度	electoral system
29	歌曲监制	producer of a song
30	作曲	composer of a song
31	成立时间	founding time
32	校歌	school song
33	擅长	be skilled in
34	亲信	trusted follower
35	前男友	ex-boyfriend
36	运动项目	sports
37	官方语言	official language

where f is the sigmoid function. The loss of logistic regression is

$$J(\boldsymbol{w}_t, \boldsymbol{x}) = -\log\left(\prod_{i=1}^{n_t} \mathrm{P}(y_t^{(i)}|\boldsymbol{x}_t^{(i)})\right), \tag{4}$$

where n_t is the number of data points, $\boldsymbol{w}_t$ is optimized iteratively using gradient descent.

Regularized multi-task learning was proposed in [6], which learns all tasks simultaneously, and gets the common sub-model shared by all the tasks as well as specific sub-models owned by each task privately. Let $\dot{h}_t$ be the hyperplane for the tth task, then $\dot{h}_t$ can be computed as Eq.(5).

$$\dot{h}_t(\boldsymbol{x}_t^{(i)}) = \boldsymbol{w}_t^\top \boldsymbol{x}_t^{(i)} \tag{5}$$

The framework of Regularized Multi-task Logistic Regression (RMTLR) is shown in Fig. 3. RMTLR assumes that $\boldsymbol{w}_t$ consists of two parts, which is shown as Eq. (6),

$$\boldsymbol{w}_t = \boldsymbol{\theta}_t + \boldsymbol{\theta}_0, \tag{6}$$

where $\boldsymbol{\theta}_0$ is the common model parameter for all tasks, and $\boldsymbol{\theta}_t$ is the specific parameter for each individual task, which represents the different characteristics of each task. The loss function of RMTLR is shown in Eq.(7), which consists of three parts,

$$\min_{\boldsymbol{\theta}_0, \boldsymbol{\theta}_t, \varepsilon_t} J(\boldsymbol{\theta}_0, \boldsymbol{\theta}_t, \varepsilon_t) = \sum_{t=1}^{T} \varepsilon_t + \frac{\lambda_1}{T}\sum_{t=1}^{T} \|\boldsymbol{\theta}_t\|^2 + \lambda_2\|\boldsymbol{\theta}_0\|^2, \tag{7}$$

where

$$\varepsilon_t = \ell(\boldsymbol{w}_t^\top \boldsymbol{x}_t), \quad \boldsymbol{x}_t = [\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)}, \ldots, \boldsymbol{x}_t^{(n)}], \tag{8}$$

T is the number of all the tasks, ε_t is the loss of logistic regression of the tth task, while the second term and the third term on the right side of Eq.(7) are regularized terms.

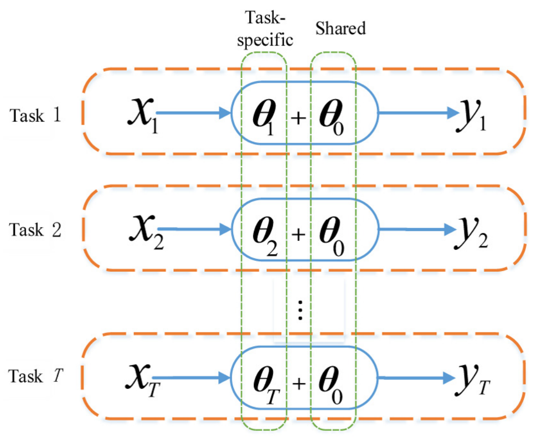

Fig. 3. The framework of RMTLR.

2.3. *Clustering by fast search and find of density peaks*

In our framework, we determine the proper task groups by a clustering method called clustering by fast search and find of density peaks [13].

The algorithm assumes that cluster centers are surrounded by neighbors with lower local density and they are at a relatively large distance from any points with a higher local density. To find the cluster centers, we need to compute two quantities for every instance: its local density ρ_i and its distance δ_i from points of higher density. Note that every data point is represented as a vector before clustering. There are many kernel methods to compute ρ_i. In this paper, ρ_i is computed with a gaussian kernel, as is shown in Eq.(9),

$$\rho_i = \sum_{j \neq i} \exp\left(-\left(\frac{d_{ij}}{d_c}\right)^2\right), \tag{9}$$

d_{ij} is the distance between data points i and j, d_c is a distance parameter. In this paper, we sort all the d_{ij} in an ascending order and set d_c as the one at the 10% position of all the distances. The smaller d_{ij} is, the greater contribution it can make to ρ_i, which means that nearby data points have more significant influence on ρ_i. The distance can be measured in a variety of manners, and we use the cosine distance in this paper.

δ_i is measured by computing the minimum distance between the point i and any other point with higher density, as is shown in Eq.(10),

$$\delta_i = \min_{j:\rho_j > \rho_i} d_{ij}. \tag{10}$$

For the point with highest density, we conventionally compute δ_i as Eq.(11),

$$\delta_i = \max_j d_{ij}. \tag{11}$$

The points with anomalously large ρ_i and δ_i values are recognized as cluster centers. Specifically, the formal definition of cluster centers are defined as Eq.(12), i.e., cluster centers are recognized as the points whose values of ρ_i and δ_i are greater than twice of the average of all the points,

$$\rho_i \geq 2 * \frac{1}{N}\sum_{j=1}^{N}\rho_j, \quad \delta_i \geq 2 * \frac{1}{N}\sum_{j=1}^{N}\delta_j, \tag{12}$$

where N is the total number of data points.

As we only use this clustering algorithm to find cluster centers, thus here we do not give the detailed steps on how to assign each data point to its cluster.

3. Framework description

3.1. Problem definition

In this section, we formally define the two problems arisen in Section 1.

3.1.1. Definition of problem 1

Given a $< s, p, o >$ triple and its corresponding sentence, our goal is to verify the correctness of $< s, p, o >$ triple with limited seed instances. The problem is whether MTL algorithms can outperform STL ones in the problem of $< s, p, o >$ mining with limited seed instances.

We treat the $< s, p, o >$ correctness verification problem for each relation as a single task, then there are totally T tasks named $\mathcal{T}_1$, $\mathcal{T}_2$, …, $\mathcal{T}_T$ (In this paper, $T = 37$). Our goal is to learn an extractor $\mathcal{E}$, which leverages the data from related tasks to promote the performance of each individual task.

3.1.2. Definition of problem 2

As we know, determining proper task groups is a fatal problem in MTL, which can directly influence the final results [9,10]. So our goal is to find related tasks to form proper task groups.

Formally, we consider the problem as the following setting. For the T tasks $\mathcal{T}_1$, $\mathcal{T}_2$, …, $\mathcal{T}_T$, our goal is to find such task group $\mathcal{G}$, in which tasks are highly related, and MTL algorithms running on $\mathcal{G}$ can get better results.

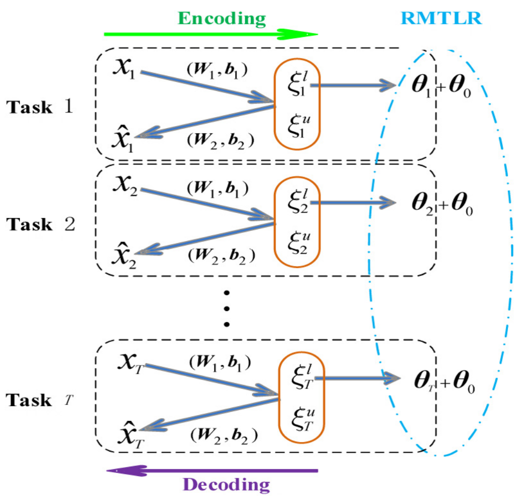

Fig. 4. The framework of S^{2}AMT.

3.2. Framework formalization

3.2.1. Framework of problem 1

Before training the models, we first transform each $< s, p, o >$ triple and its corresponding sentence into a feature vector, then the problem of $< s, p, o >$ correctness verification can be regarded as a binary classification problem. We totally extracted 213 features to represent each $< s, p, o >$ triple and its corresponding sentence, including 26 semantic features and 187 structural ones. The features are given in Section 4.2.

Then we propose the S^{2}AMT model to solve the problem. The framework of S^{2}AMT is shown in Fig. 4, which consists of two substeps, an autoencoder step and a RMTLR step. We introduce the details of these two substeps in the following section.

3.2.1.1. Autoencoder. S^{2}AMT utilizes autoencoder to obtains the latent representation of input data. In the autoencoder step, S^{2}AMT makes full use of not only the labeled but also the unlabeled instances. In Fig. 4, task 1 to task T belong to the same task group. $\boldsymbol{x}_t \in \mathbb{R}^{m\times(n_{tu}+n_{tl})}$ means the input of the tth task while $\widehat{\boldsymbol{x}}_t \in \mathbb{R}^{m\times(n_{tu}+n_{tl})}$ is the corresponding output, where n_{tl} is the number of labeled data in task t and n_{tu} is the corresponding number of unlabeled data, m is the number of dimensions of the input and output layers. $\boldsymbol{W}_1 \in \mathbb{R}^{k\times m}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{k\times 1}$ are the parameters for encoding while $\boldsymbol{W}_2 \in \mathbb{R}^{m\times k}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{m\times 1}$ are the corresponding parameters for decoding, where k is the number of dimensions of the hidden layer. $\boldsymbol{\xi}_t^l \in \mathbb{R}^{k\times n_{tl}}$ and $\boldsymbol{\xi}_t^u \in \mathbb{R}^{k\times n_{tu}}$ represent the hidden layer for the labeled and unlabeled data respectively, and $\boldsymbol{\xi}_t = \boldsymbol{\xi}_t^l \cup \boldsymbol{\xi}_t^u$. With the autoencoder, the intrinsic knowledge of unlabeled data can be utilized.

3.2.1.2. RMTLR. In this step, the RMTLR model is trained based on the latent representations of the labeled data from the tasks within the same group. $\boldsymbol{\theta}_t \in \mathbb{R}^{k\times 1}$ and $\boldsymbol{\theta}_0 \in \mathbb{R}^{k\times 1}$ are the parameters of RMTLR, where $\boldsymbol{\theta}_0$ is the common model parameter for tasks in the same group, and $\boldsymbol{\theta}_t$ is the task-specific parameter. In this way, every task can use the information not only from itself but also from other tasks.

3.2.1.3. Loss function. The input of S^{2}AMT is T task data sets with both labeled and unlabeled data, i.e., $\boldsymbol{D}_t = \{\boldsymbol{x}_t^{(i)}, y_t^{(i)}\}_{i=1}^{n_{tl}} \cup \{\boldsymbol{x}_t^{(i)}\}_{i=1}^{n_{tu}}$ $(1 \leq t \leq T)$, where $\boldsymbol{x}_t^{(i)} \in \mathbb{R}^{m\times 1}$, $y_t^{(i)} \in \{0, 1\}$. Note that every $< s, p, o >$ triple and its corresponding sentence has been represented as a feature vector.

The loss function of S²AMT is

$$\mathcal{L}=\sum_{t=1}^{T}\mathcal{L}_{AE}(\boldsymbol{x}_t,\hat{\boldsymbol{x}}_t)+\alpha\sum_{t=1}^{T}\mathcal{L}_{LR}(\boldsymbol{\xi}_t,\hat{\boldsymbol{\theta}}_t)+\Omega_{AE}+\Omega_{LR}. \tag{13}$$

$\mathcal{L}_{AE}(\boldsymbol{x}_t,\hat{\boldsymbol{x}}_t)$ is the reconstruction error of autoencoder, which is defined as follow,

$$\mathcal{L}_{AE}(\boldsymbol{x}_t,\hat{\boldsymbol{x}}_t)=\sum_{i=1}^{n_t}\left\|\hat{\boldsymbol{x}}_t^{(i)}-\boldsymbol{x}_t^{(i)}\right\|^2, \tag{14}$$

where

$$\begin{aligned} n_t &= n_{tl}+n_{tu},\\ \boldsymbol{\xi}_t^{(i)} &= f(\boldsymbol{W}_1\boldsymbol{x}_t^{(i)}+\boldsymbol{b}_1),\\ \hat{\boldsymbol{x}}_t^{(i)} &= f(\boldsymbol{W}_2\boldsymbol{\xi}_t^{(i)}+\boldsymbol{b}_2), \end{aligned} \tag{15}$$

f is the sigmoid function.

$\sum_{t=1}^{T}\mathcal{L}_{LR}(\boldsymbol{\xi}_t,\hat{\boldsymbol{\theta}}_t)$ is the optimization problem of multi-task logistic regression, which is the negative value of the log likelihood,

$$\begin{aligned} &\mathcal{L}_{LR}(\boldsymbol{\xi}_t,\hat{\boldsymbol{\theta}}_t)\\ &=-\log\left(\prod_{i=1}^{n_{tl}}\mathrm{P}(y_t^{(i)}|\boldsymbol{\xi}_t^{(i)},\hat{\boldsymbol{\theta}}_t)\right)\\ &=-\log\left(\prod_{i=1}^{n_{tl}}(f(\hat{\boldsymbol{\theta}}_t^\top\boldsymbol{\xi}_t^{(i)}))^{y_t^{(i)}}(1-f(\hat{\boldsymbol{\theta}}_t^\top\boldsymbol{\xi}_t^{(i)}))^{1-y_t^{(i)}}\right)\\ &=-\sum_{i=1}^{n_{tl}}\left(y_t^{(i)}\log f(\hat{\boldsymbol{\theta}}_t^\top\boldsymbol{\xi}_t^{(i)})+(1-y_t^{(i)})\log(1-f(\hat{\boldsymbol{\theta}}_t^\top\boldsymbol{\xi}_t^{(i)}))\right), \end{aligned} \tag{16}$$

where

$$\hat{\boldsymbol{\theta}}_t=\boldsymbol{\theta}_t+\boldsymbol{\theta}_0,\ \hat{\boldsymbol{\theta}}_t\in\mathbb{R}^{k\times 1}, \tag{17}$$

If $f(\hat{\boldsymbol{\theta}}_t^\top\boldsymbol{\xi}_t^{(i)})>0.5$, $\boldsymbol{\xi}_t^{(i)}$ belongs to class 1, otherwise $\boldsymbol{\xi}_t^{(i)}$ belongs to class 0.

Ω_{LR} is the regularized term of multi-task logistic regression, which is defined as follow,

$$\Omega_{LR}=\frac{\lambda_1}{2T}\sum_{t=1}^{T}\|\boldsymbol{\theta}_t\|^2+\frac{\lambda_2}{2}\|\boldsymbol{\theta}_0\|^2, \tag{18}$$

where λ_1 and λ_2 are trade-off parameters. Large value of λ_1 will lead to all tasks sharing the same model, while large value of λ_2 will result in the separate training of each task.

Ω_{AE} is the regularized term of the autoencoder, which is defined as follow,

$$\Omega_{AE}=\lambda_3(\|\boldsymbol{W}_1\|^2+\|\boldsymbol{b}_1\|^2+\|\boldsymbol{W}_2\|^2+\|\boldsymbol{b}_2\|^2), \tag{19}$$

where λ_3 is a regularized parameter.

With the loss function Eq.(13), we use gradient descent to solve the problem, and derive the solutions of $\boldsymbol{W}_1,\boldsymbol{b}_1,\boldsymbol{W}_2,\boldsymbol{b}_2,\boldsymbol{\theta}_t,\boldsymbol{\theta}_0$. The details of solution derivation can be referred in the Appendix . Note that in order to get the best performance of our framework, we initialize $\boldsymbol{\theta}_t$ as the weight of single task logistic regression and $\boldsymbol{\theta}_0$ is the average of all $\boldsymbol{\theta}_t$.

3.2.2. Framework of problem 2

In order to find proper task groups, we first get the representation vector of each task and run the clustering algorithm on the representation vectors.

3.2.2.1. Getting representation vectors. The process of getting representation vectors is shown in Fig. 5. At the beginning, the $<s,p,o>$ triples and their corresponding sentences are transformed into feature vectors using the features described in Section 4.2.

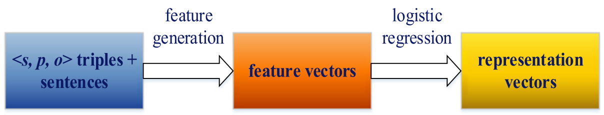

Fig. 5. The process of getting feature vectors.

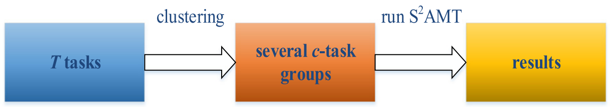

Fig. 6. The process of getting feature vectors.

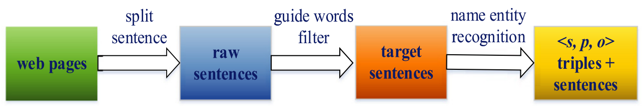

Fig. 7. The process of getting data set.

After that, we run logistic regression for each task with labeled data, and get a hyperplane for each task to separate the positive data points from the negative ones. The hyperplane for task t is $\dot{h}_t$, which is represented as Eq.(5), and $\boldsymbol{w}_t$ represents the vector for the tth task.

3.2.2.2. Task clustering. We run the clustering algorithm described in Section 2.3 on T representation vectors $\boldsymbol{w}_1,\boldsymbol{w}_2,\ldots,\boldsymbol{w}_T$, and obtain cluster centers for each task group. The center task and its nearest $c-1$ tasks form a c-task group, then we run S²AMT on them to get the final results. The overall process is shown in Fig. 6.

Note that we use the cosine similarity to measure the distance between representation vectors. For any pair of tasks, the more similar they are, the smaller distance between their corresponding representation vectors is.

4. Experiments

4.1. Data set

We prepare the data set using the Baidu search engine. Moreover, we impose heuristic constraints for each relation. For example, the s and o of the relation "老乡" ('fellow-townsman') should be both PER (person) type entities, the s of the relation "作曲者" ('composer of a song') should be a SNG (song) type entity and the o should be a PER type entity.

Before introducing the process of getting the data set, we present the definitions of guide word, guide word filter and s/o candidate.

- Guide word. Every relation has at least one guide word to detect whether a sentence has $<s,p,o>$ triples of this relation. Sentences that contain at least one guide word of a relation may probably imply a knowledge triple of the relation. For example, the relation "妻子" ('wife') has guide words "妻子" ('wife'), "丈夫" ('husband') and "结婚" ('marry'); the relation "女儿" ('daughter') has guide words "女儿" ('daughter'), "父亲"('father') and "母亲" ('mother'). We collect the guide words for each relation manually and get a close set of guide words.
- Guide word filter. A guide word filter is basically a heuristic rule. That is, if a sentence contains a guide word of a relation, the sentence becomes a target sentence of this relation.

Table 2
# of seed instances in training data.

# of seed instances	# of tasks
>=10 and <=100	22
>100 and <=500	9
>500	6

- s/o candidate. All the entities in the sentence that meet the entity type constraints of p for s/o become s/o candidates. For example, in a sentence of the relation "作曲者" ('composer of a song'), all the SNG type entities become the candidates of s while all the PER type entities become the candidates of o.

As shown in Fig. 7, we process the data set by three steps. First, we split the free text in web pages into raw sentences. Next, we use guide word filters to get target sentences for each p. With the guide word filter, the number of raw sentences to target sentences decreases sharply. Finally, we run name entity recognition on these target sentences.[4] If the types of name entities can meet the entity type requirements of the target relation, entities and the target relation can form a $< s, p, o >$ triple. Taking the sentence in Fig. 1 as an example, after name entity recognition, we get PER entities Barack Obama, Michelle Obama, Sasha Obama and Malia Obama. As the relation *wife* requires s and o should be both PER type entities, we can get candidate $< s, p, o >$ triples like <Barack Obama, wife, Michelle Obama> and <Barack Obama, wife, Sasha Obama>, etc.

In this way, our data set can be split into 37 sub-datasets with respect to the 37 relations according to the guide words for each relation. Each instance contains a $< s, p, o >$ triple and its corresponding sentence. And each instance is manually annotated with a label, i.e., True or False. We randomly split the data into training data and test data, leaving 1000 seed instances of each relation in the test data. The number of seed instances in training data is shown in Table 2. Note that negative instances in both training and test data are sufficient. From Table 2 we can see that most tasks have less than 100 seed instances, which are quite small.

4.2. Feature construction

We extract 213 features to represent each $< s, p, o >$ triple and its corresponding sentence. These features are divided into two groups. One group is semantic features, which contains 26 features, and the other group is structural features, which contains 187 features. Here we only give 5 structural features and 5 semantic ones. The details of the 213 features are given in the Appendix .

First, let us explain 5 structural features.

- The number of Chinese characters between s and the guide word of p. Similarly, we have features like the number of Chinese characters between the guide word and o, the number of Chinese characters between s and o, etc.
- The number of nouns between s and the guide word of p. Similarly, we have features like the number of nouns between the guide word and o, the number of personal pronouns between s and the guide word, etc.
- The number of entities in the sentence that belong to the same entity type as s.
- Whether the guide word of the relation is the closest guide word to the s, if true it is 1, otherwise 0.
- Whether the s of the $< s, p, o >$ triple lies ahead of the o. The value is 1 if the s lies ahead of the o, otherwise 0.

[4] The name entity recognition tool we use is developed by Baidu, which use max entropy model.

Then, we also take 5 semantic features to explain.

- The number of occurrences of queries that contain s and p in one year's query logs. The intuition is that the more the number of times this pair is searched, the more likely it is a true pair. The query logs we use are from Baidu.
- Whether o is unique under the relation. The value is 1 if it is unique, otherwise 0. For example, under the relation "成立时间" ('founding time'), o is unique, while under the relation "爱好" ('hobbies'), o is not unique.
- The similarity between $s + p$ and o. We train a model to compute the similarity of two queries. If the URLs that the two queries lead to in query logs are highly coincident, the two queries have a high similarity according to the model. The training data for this model is one year's query logs from Baidu, and a neural network is used as the training model.
- The distance between the o and the nearest negative word, which is measured by the number of Chinese characters. The distance is regarded as 50 if there is no negative word in the sentence.
- Whether there is a negative word in the sentence, if true it is 1, otherwise 0.

4.3. Baselines

4.3.1. Baselines for problem 1

To demonstrate the effectiveness of the proposed algorithm, we compare our model against the following baselines.

- Ridge regression (RR) [14] : a classifier that is based on the least squares of input data.
- Support vector machine (SVM) [15] : a robust classifier which aims to find a maximum-margin hyperplane lying between the positive instances and the negative ones.
- Logistic regression (LR) [12] : a classifier that is used to estimate the probability of a binary response.
- Exploring Various Knowledge in Relation Extraction (EVKRE) [16] : a feature-based relation extraction method which gives an overall analysis of efficient features.
- Dynamic Syntactic Parse Tree (DSPT) [17] : a kernel-based relation extraction method.
- Guided Distant Supervision (Guided DS) [18]: a distant IE method which uses both the human-labeled information and distant supervision from KBs.
- RMTLR [6] : a supervised MTL algorithm based on LR. Different from S^{2}AMT, this algorithm does not have the autoencoder step and only uses supervised information.
- LSTM-RNN [19]: an end-to-end relation extraction algorithm, which extracts relations between entities based on word sequence information and dependency tree structures.
- Global Optimized Relation Extraction (GlobalRE) [20]: another end-to-end relation extraction algorithm, which makes use of global normalization and syntactic features to conduct relation extraction.
- oblique Random forest with random vector Functional Link network (obRaFL) [21] : a multi-class classification model which combines random forests with deep neural networks. Instead of treating the knowledge triple mining problem as multiple binary classification problems, obRaFL regards the problem as a multi-class classification problem with 38 class labels, i.e., 37 relations as 37 classes and 1 extra class for the null class.

4.3.2. Baselines for problem 2

In Problem 2, we try to find some task groups in which the tasks are semantically similar. To evaluate the performance, we adopt the following two baselines as well as the ones described in Section 4.3.1.

- S^2AMT for 37 tasks : this baseline refers to that we run S^2AMT on all the 37 tasks as one task group.
- RMTLR for a task group : this baseline refers to that we run RMTLR on a given task group.

4.4. Metrics and experimental settings

Average precision, average recall and average F1-score are adopted to evaluate all the algorithms, and they are respectively denoted as P, R and F. P_t, R_t and F_t are used to denote the precision, recall, and F1-score of the tth task, then

$$P = \frac{1}{T}\sum_{t=1}^{T} P_t, \quad R = \frac{1}{T}\sum_{t=1}^{T} R_t, \quad F = \frac{1}{T}\sum_{t=1}^{T} F_t. \tag{20}$$

The experimental settings are shown as follows,

- For all the parameters in all the algorithms, we use a 5-fold cross-validation on the training set to select the best parameters.
- The paper of EVKRE [16] gives an overall analysis of efficient features. We adopt the most efficient feature combination in [16] to represent each $< s, p, o >$ triple and its corresponding sentence instead of the features described in Section 4.2.
- For DSPT [17], we use the same features and kernel as the original work rather than using the features described in Section 4.2.
- For Guided DS [18], we manually annotate half of the training instances and use the distant supervision from Baidupedia[5] to annotate the rest with the method in Guided DS.
- For LSTM-RNN, we adopt the shortest path tree version of the model, which is denoted as LSTM-RNN (SPTree) in the original work.

For all the methods except Guided DS and LSTM-RNN, we implement the methods by ourselves. For Guided DS, we implement Guided DS based on the MIML [22] code.[6] For LSTM-RNN, we slightly change the code[7] to fit our data format.

4.5. Experimental results

4.5.1. Results of problem 1

In order to investigate whether MTL methods can outperform STL ones, we assign all the 37 tasks to only one group and run MTL methods on this group. All the results are shown in Table 3, and R, P, F denote the average recall, precision and F1-score respectively. Note that, RMTLR and S^2AMT are MTL methods while the others are STL methods.

From Table 3, we have the following observations. RMTLR and S^2AMT outperform STL methods. Specifically, RMTLR obtains the highest average precision while S^2AMT achieves the highest average recall and highest F1-score, which indicates that MTL methods have better performance than STL ones with limited seed instances. We also find that deep learning based models like LSTM-RNN, GlobalRE and obRaFL cannot perform very well due to the lack of enough seed instances for training.

Using the MTL methods, some tasks can obtain the values of F1-score higher than 90%, e.g., the F1-score of “专业代码” (‘subject ID’) and “身高” (‘body height’) are 97.26% and 96.30%. In this case, they can be added to the Baidu KB to support downstream applications.

[5] https://baike.baidu.com/.
[6] Available at http://nlp.stanford.edu/software/mimlre.shtml.
[7] Available at https://github.com/tticoin/LSTM-ER.

Table 3
The results of all the 37 tasks.

	Algorithm	R	P	F
STL methods	RR	50.45	62.95	51.69
	SVM	49.94	62.04	51.09
	LR	48.19	64.85	52.87
	EVKRE	51.68	65.39	53.38
	DSPT	50.96	66.39	53.02
	Guided DS	49.69	70.48	52.69
	LSTM-RNN	23.87	41.32	30.31
	GlobalRE	22.67	45.31	29.86
	obRaFL	19.63	40.12	27.18
MTL methods	RMTLR	47.80	**71.24**	53.92
	S^2AMT	**53.02**	63.23	**54.39**

Table 4
The results of SNG type relations.

	Algorithm	R	P	F
STL methods	RR	53.23	32.48	40.28
	SVM	48.21	36.06	40.68
	LR	41.07	38.91	39.59
	EVKRE	45.63	43.26	43.93
	DSPT	46.96	46.39	46.51
	Guided DS	50.33	39.65	42.39
	LSTM-RNN	25.32	24.96	25.27
	GlobalRE	27.55	25.46	26.50
	obRaFL	22.31	20.29	21.96
MTL methods	S^2AMT for 37 tasks	**70.98**	41.22	**50.01**
	RMTLR for PER tasks	52.23	**47.32**	46.79
	S^2AMT for PER tasks	52.23	41.67	43.58

Table 5
The results of PER type relations.

	Algorithm	R	P	F
STL methods	RR	36.83	66.69	47.07
	SVM	42.46	70.07	48.89
	LR	36.49	68.68	46.47
	EVKRE	39.62	69.32	48.53
	DSPT	38.91	**72.69**	48.32
	Guided DS	41.36	71.30	47.65
	LSTM-RNN	22.63	30.32	23.28
	GlobalRE	21.19	29.22	22.83
	obRaFL	20.39	27.36	20.91
MTL methods	S^2AMT for 37 tasks	**42.92**	69.70	**48.96**
	RMTLR for SNG tasks	36.37	69.22	45.72
	S^2AMT for SNG tasks	42.02	61.27	48.17

4.5.2. Experimental results of problem 2

We try to find some proper task groups to further improve the performance. Intuitively, we think some relations which have the same constraints on s and o should be grouped together. For example, the relation “老乡” (‘fellow-townsman’) and “前男友” (‘ex-boyfriend’) have the same constraints that the s and o should be both PER type entities. In this paper, we find 8 relations which meet the requirement that the s and o are both PER type entities. We call these relations PER type relations, and the corresponding $< s, p, o >$ mining tasks are called PER tasks. Similarly, both the relations “歌曲监制” (‘producer of a song’) and “作曲者” (‘composer of a song’) require s should be SNG type and o should be PER type. We call these relations SNG type relations, and the corresponding tasks are called SNG tasks. Can PER tasks and SNG tasks form the proper task groups? To answer this question, we run some experiments on PER tasks and SNG tasks, and the results are shown in Tables 4 and 5. S^2AMT for 37 tasks refers to that we run our method on all the 37 tasks as one group and get the results of the corresponding PER/SNG tasks.

From Tables 4 and 5, we find that for both PER and SNG type relations, RMTLR and S^2AMT for PER/SNG tasks cannot outperform S^2AMT for 37 tasks on F. This indicates that seemingly similar

Table 6
The results of SNG type relations with Semantic features.

	Algorithm	*R*	*P*	*F*
STL methods	RR	36.84	32.86	33.37
	SVM	32.14	16.67	21.95
	LR	35.27	31.67	32.51
	EVKRE	37.21	20.98	23.91
	DSPT	36.98	31.36	32.33
	Guided DS	38.29	32.46	34.16
	LSTM-RNN	25.32	24.96	25.27
	GlobalRE	27.55	25.46	26.50
	obRaFL	19.29	18.56	18.99
MTL methods	S^2AMT for 37 tasks	36.86	17.14	24.49
	RMTLR for PER tasks	38.43	33.64	34.67
	S^2AMT for PER tasks	**40.63**	**37.08**	**37.31**

Table 7
The results of PER type relations with Semantic features.

	Algorithm	*R*	*P*	*F*
STL methods	RR	27.07	48.66	33.24
	SVM	28.99	45.50	34.22
	LR	31.01	54.01	40.61
	EVKRE	34.27	56.31	39.81
	DSPT	33.56	51.36	40.66
	Guided DS	33.48	55.48	40.32
	LSTM-RNN	22.63	30.32	23.28
	GlobalRE	21.19	29.22	22.83
	obRaFL	17.39	17.66	17.40
MTL methods	S^2AMT for 37 tasks	33.56	48.13	36.09
	RMTLR for SNG tasks	35.49	52.09	41.01
	S^2AMT for SNG tasks	**37.87**	**61.25**	**44.36**

tasks may not be able to form an appropriate task group. After observing the sentences of PER/SNG type relations, we find that sentences corresponding to different relations may have different structural features. Moreover, the diversity of guide words makes this phenomenon even more serious. Here we give an example as follows,

- Sentence 1: 他甚至不知道，莱奥纳多・迪卡普里奥(o) 是芭儿・拉法莉(s) 的前男友(p)。(He does not even know that Leonardo DiCaprio (o) is Bar Refaeli's (s) ex-boyfriend (p).) Triple: < 芭儿・拉法莉 (Bar Refaeli) 前男友, (ex-boyfriend), 莱奥纳多・迪卡普里奥 (Leonardo DiCaprio)>
- Sentence 2: 在科比・布莱恩特(s) 的职业生涯中,他曾经与许多优秀的球员合作,包括卡尔・马龙、保罗• 加索尔以及与他传闻不和(p) 的沙奎尔• 奥尼尔(o)。(In Kobe Bryant's (s) career, he has played with many excellent players, including Karl Malone, Pau Gasol and Shaquille O'Neal (o), whom it is said he has had a bad relationship with (p).) Triple: < 科比・布莱恩特 (Kobe Bryant), 传闻不和 (rumored to have a bad relation with) 沙奎尔・奥尼尔 (Shaquille O'Neal)>

These two sentences differ in the structural features like the relative position of s, p and o, and the number of PER type entities between s and o, etc. Therefore, we guess that the difference of structural features lead to the failure of PER/SNG tasks to form proper task groups. If we only use semantic features, whether PER tasks and SNG tasks can form proper task groups? To verify this conjecture, we perform the experiments only using semantic features, and the results are shown in Tables 6 and 7.

From Tables 6 and 7, we have the following findings.

- For PER and SNG task groups, S^2AMT for PER/SNG tasks obtains the best results according to recall, precision and F1-score, and RMTLR for PER/SNG tasks obtains the runner-up results on F1-score. This indicates that if we only use semantic features, PER tasks and SNG tasks can form proper task groups.

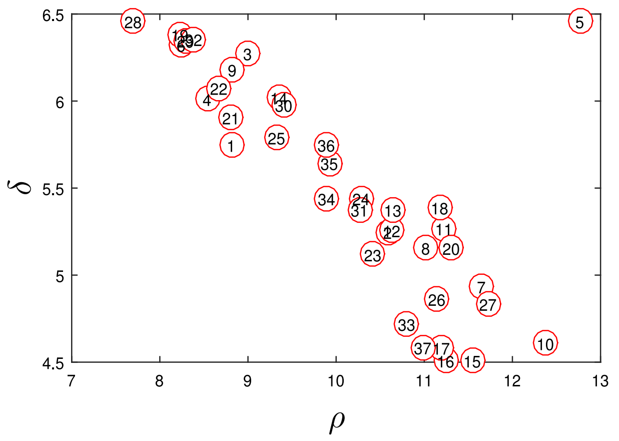

Fig. 8. Decision graph.

- Comparing the results in Tables 6, 7 with the ones in Tables 4, 5, we find that better results can be obtained when both structural and semantic features are used, which indicates the usefulness of structural features to promote the performance.

Since PER and SNG tasks cannot form proper task groups using all the 213 features, so we use a cluster method to find some proper task groups. First, we run logistic regression on each tasks and get the representation vectors for each task. Then, we run the clustering method introduced in Section 2.3 based on these representation vectors to find cluster centers. The decision graph is shown in Fig. 8, the horizontal axis represents the local density ρ, while the vertical axis represents the distance δ from the points of higher density.

The values of ρ and δ should be large for centers, so they locate at the top right corner of the graph. From Fig. 8 we find one cluster center, and the corresponding relation is the "专业代码" ('subject ID'). Our target is to find c-task groups. Larger value of c leads to more tasks forming one group, while some unrelated tasks may join the same group. Smaller value of c leads to less tasks joining the same group, thus less information can be leveraged between each task. We use a 5-fold cross-validation on the training set to select the best value of c from $\{3, 4, \ldots, 10\}$, finally c is set to 5.

Finally, we run all algorithms on this 5-task group, and the results are shown in Table 8. S^2AMT for 37 tasks refers to that we run our method on all the 37 tasks as one group and get the results of the selected 5 tasks. S^2AMT for 37 tasks differs from S^2AMT for 5 tasks in that the former one does not have the clustering step. RMTLR for 5 tasks differs from S^2AMT for 5 tasks in that the former one does not adopt autoencoder, thus cannot leverage the information from the unlabeled data. From Table 8, we have the following observations:

- S^2AMT for 5 tasks outperforms S^2AMT for 37 tasks, which indicates the importance of finding proper task groups. The results also show that the clustering method we use can find proper task groups to further improve the performance.
- S^2AMT and RMTLR for 5 tasks outperform STL methods, which verifies again that MTL methods can promote the performance of the knowledge triple mining problem.
- S^2AMT for 5 tasks outperforms RMTLR for 5 tasks, which indicates the importance of learning from a large amount of unlabeled data.

4.5.3. *The Influence of the Number of Seed Instances on S^2AMT*

From Table 2, there are 15 tasks that have more than 100 seed instances in training data, then we run all the algorithms on these

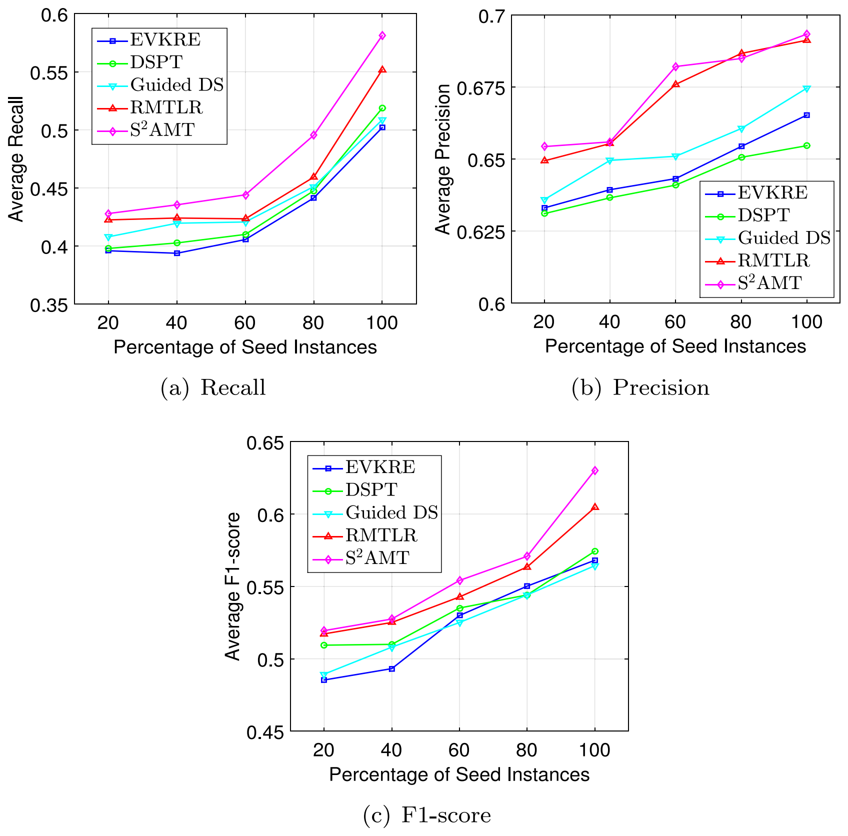

Fig. 9. The change of results with the number of seed instances increasing.

Table 8
The results of the 5-task Group.

	Algorithm	*R*	*P*	*F*
STL methods	RR	40.35	42.69	41.04
	SVM	36.35	41.21	38.16
	LR	41.88	50.71	44.89
	EVKRE	44.68	59.32	47.63
	DSPT	45.32	58.45	46.92
	Guided DS	42.36	57.37	47.45
	LSTM-RNN	28.36	35.16	31.49
	GlobalRE	29.64	33.61	31.23
	obRaFL	25.91	37.69	30.10
MTL methods	S^{2}AMT for 37 tasks	42.35	61.43	46.55
	RMTLR for 5 tasks	45.47	62.67	48.61
	S^{2}AMT for 5 tasks	**48.36**	**80.00**	**53.90**

Table 9
K-Means results.

Algorithm	*R*	*P*	*F*
k-means+RMTLR (Group 1)	46.10	59.61	48.39
LR (Group 1)	48.36	61.32	50.49
k-means+RMTLR (Group 2)	49.61	65.34	57.96
LR (Group 2)	55.96	70.39	59.25
k-means+RMTLR (Group 3)	43.26	64.69	48.62
LR (Group 3)	40.91	60.86	45.26
k-means+RMTLR (Group 4)	72.36	57.23	65.01
LR (Group 4)	74.62	59.37	65.92
k-means+RMTLR (Group 5)	43.42	68.22	48.33
LR (Group 5)	44.62	71.99	49.39
k-means+RMTLR (Group 6)	49.53	64.30	54.16
LR (Group 6)	51.22	79.62	59.42
k-means+RMTLR (overall)	46.39	60.02	50.11
LR (overall)	**48.19**	**64.85**	**52.87**

15 tasks with the percentage of seed instances from 20% to 100%. That is, we first add 20% of the seed instances to training data, then 40%, 60%, etc. The results are shown in Figs. 9(a)–9(c), and we find that,

- S^{2}AMT and RMTLR outperform other STL methods, which indicates that MTL algorithms have advantages over STL ones.
- MTL algorithms perform better with the increasing number of seed instances, which shows MTL algorithms can benefit from different numbers of seed instances.
- Generally, S^{2}AMT outperforms RMTLR, which shows the importance of learning from unlabeled data.

Note that in the beginning, we tried to use k-means algorithm to obtain task groups. We run k-means on the 37 representation vectors to find task groups and run the MTL algorithm RMTLR on these groups. This setting differs from S^{2}AMT only in the clustering algorithm. k is selected from $\{3, 4, 5, \dots, 10\}$ and the best results fall when $k = 6$. To test the quality of this setting, we also run Logistic Regression (LR) on each task.

The results are shown in Table 9. We can see that for these task groups, generally k-means+RMTLR cannot outperform LR. We believe the reason lies in as follow: the quality of the task groups obtained by k-means cannot be guaranteed. K-means assigns each task to a task group. However, there exist some tasks that do not have much similarity with other tasks and MTL methods are not suitable to solve these tasks, which will affect the overall performance of a task group. When the number of tasks in a group is small, this kind of tasks play a more important role, making the problem even more serious.

5. Related work

Traditional supervised IE methods can be divided into two categories, feature-based [23,24] and kernel-based [17,25] methods. Feature-based methods use a set of useful features to train classifiers, while kernel-based ones provide a natural alternative to exploit rich representations of the input classification clues, such as syntactic parse trees [26]. Recently, two kinds of supervised IE methods attracted people's attention. The first kind is distant IE [18,27], which gets distant supervised information from existing KBs like Freebase [28]. The second kind is based on deep learning approaches [19,29,30], which leverages the advantages of deep neural networks in understanding languages. The combination of the two kinds of methods is also a promising area [31,32]. However, the existing methods have the following shortcomings. First, the availability of sufficient seed instances is always assumed before training models, which does not always hold in practice [5]. Second, most methods only use supervised information, while ignoring a large amount of unlabeled data, which may also contain plenty of useful information.

MTL conducts multiple related learning tasks simultaneously so that the useful information in one task can be used for other tasks, and the performance of every individual task can be promoted. Generally, there are two types of MTL methods. One kind is using the supervised information from all the tasks to help tasks to do feature selection collectively or learn a shared feature space for knowledge sharing [33–36]. The other kind of methods focus on sharing model parameters among multiple tasks based on the original features [6–8,37]. Since MTL methods can use the information from other tasks, they can often outperform STL methods [6–8]. Besides, many MTL algorithms use not only supervised information but the unsupervised information learned from unlabeled data [38,39], and S^2AMT is one of them. S^2AMT uses autoencoder to get the latent representation of each input instances, thus leveraging the intrinsic information of unlabeled data.

Though many supervised IE and MTL methods have been proposed, the combination of the two areas remains pretty much open. To the best of our knowledge, the most related work [5] assumes that only the target task has limited seed instances while other tasks have plenty of seed instances. Our work assumes that most tasks have limited seed instances, and our model is capable to promote the performance of all tasks simultaneously.

6. Conclusion

In this paper, we proposed a MTL method S^2AMT to solve the problem of knowledge triple mining with limited seed instances, and the experiments on real-world data sets demonstrated the effectiveness of the MTL model over STL ones. Moreover, we provided a clustering algorithm to find proper task groups to further improve the performance. We also had the insightful observation that seemingly similar tasks may not be appropriate to form a task group. Most importantly, our work provided a new perspective for knowledge triple mining when the number of seed instances is small.

Acknowledgements

The research work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB1004300, the National Natural Science Foundation of China under Grant No. 61773361, 61473273, 91546122, Guangdong provincial science and technology plan projects under Grant No. 2015 B010109005, the Project of Youth Innovation Promotion Association CAS under Grant No. 2017146.

Appendix

I Partial derivatives of optimizing parameters

In order to get better parameters of our model, we develop an alternately optimization algorithm based on partial derivatives to derive the solutions, which is shown as follows,

$$\begin{aligned}
\boldsymbol{W}_1 &\leftarrow \boldsymbol{W}_1 - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{W}_1}, & \boldsymbol{b}_1 &\leftarrow \boldsymbol{b}_1 - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{b}_1},\\
\boldsymbol{W}_2 &\leftarrow \boldsymbol{W}_2 - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{W}_2}, & \boldsymbol{b}_2 &\leftarrow \boldsymbol{b}_2 - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{b}_2},\\
\boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_t - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_t}, & \boldsymbol{\theta}_0 &\leftarrow \boldsymbol{\theta}_0 - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_0},
\end{aligned} \tag{21}$$

where η is the learning rate.

II Details of the 213 features

Here we give the details of the 213 features, which include 187 structural features and 26 semantic ones. First, let us explain structural features. For each $< s, p, o >$ triple and its sentence, the corresponding guide word of p in the sentence is denoted as g.

1. The number of guide words of other p in the sentence.
2. The number of the candidates of s in the sentence.
3. The number of the candidates of o in the sentence.
4. The number of the candidates of s/o pairs in the sentence. For example, if there are 2 candidates of s and 3 candidates of o, the number is 6.
5. Whether there is a guide word of another p between s and o.
6. Whether there is a guide word of another p within 2 Chinese characters from o.
7. Whether there is another candidate of o between o and g.
8. Whether there is another candidate of s between s and g.
9. Whether the guide word of p is the nearest guide word to o.
10. Whether the guide word of p is the nearest guide word to s.
11. Whether the guide word of p is the nearest guide word to s and o. We compare the sum of the distances of s to g and g to o, the distance we use is the number of Chinese characters (the same below).
12. The rank of the distance between this o and g in all the candidates of o.
13. The rank of the distance between this s and g in all the candidates of s.
14. The rank of the sum of the distance of s to g and g to o in all the candidates of s/o pairs.
15. The rank of the sum of the distance of s to g and s to o in all the candidates of s.
16. The rank of the sum of the distance of s to g and g to o in all the candidates of g.
17. The rank of the sum of the distance of s to o and g to o in all the candidates of o.
18. Whether there is a space between s and g.
19. Whether there is a space between s and o.
20. Whether there is a space between o and g.
21. Whether p has more than 1 guide word.
22. Whether g lies in front of s and o.
23. Whether g lies between s and o.
24. Whether g lies behind s and o.
25. The distance between s and g.
26. The distance between g and o.
27. The number of verbs between s and g.
28. The number of verbs between g and o.
29. The number of nouns between s and g.
30. The number of nouns between g and o.

31. The number of guide words of other p between s and g.
32. The number of guide words of other p between g and o.
33. The number of entities between s and g that belong to the same entity type as s.
34. The number of entities between g and o that belong to the same entity type as o.
35. The number of auxiliary words between s and g.
36. The number of auxiliary words between g and o.
37. The number of personal pronouns between s and g.
38. The number of personal pronouns between g and o.
39. The number of commas between s and g.
40. The number of commas between g and o.
41. The number of conjunctions between s and g.
42. The number of conjunctions between g and o.
43. The number of all the guide words in the sentence.
44. The number of all the entities that belong to the same entity type as s in the sentence.
45. The number of all the entities that belong to the same entity type as o in the sentence.
46. The number of guide words of other p in the sentence.
47. The distance between s and o.
48. The number of entities between s and o that belong to the same entity type as s.
49. The number of entities between s and o that belong to the same entity type as o.
50. The number of commas between s and o.
51. The number of conjunctions between s and o.
52. The number of personal pronouns between s and o.
53. The part of speech tag of the word before s.
54. The part of speech tag of the word after s.
55. The part of speech tag of the word before o.
56. The part of speech tag of the word after o.
57. The number of the negative words in the sentence.
58. The num of Chinese characters of o.
59. Whether there is a guide word of another p within 2 Chinese characters from s.
60. Whether a comma lies in front of o.
61. Whether there is a Chinese characters “是” (‘is’) between s and o.
62. The relative position of s and g.
63. The relative position of o and g.
64. The relative position of s and o.
65. The relative position of s, g and o.
66. The number of quotation marks between s and o.
67. The position of s in the sentence.
68. The position of o in the sentence.
69. The average value of all the structural features above.

For many structural features above, they can be split into 2 to 5 features. Taking the 1-st feature as an example, it can be split into 3 features: whether the number of guide words of other p in the sentence is 0, between 1 to 3, or more than 4. In this way, 69 features above are split into 187 features, which we call structural features.

Then, we also take semantic features to explain.

1. Whether the genders of s and o meet the requirements. This feature is only used under some specific relations like “前男友” (‘ex-boyfriend’). In other relations, this feature is always set to be 1. We use a tool developed by Baidu to test whether a Chinese name is a name of a male or a female, a neural network is used to train the model.
2. Whether there is a containment relationship between s and o. Here we use string-level containment relationship, for example, ‘Beijing’ is contained in ‘the city of Beijing’.
3. Whether s has the same family name as o, this feature is only used under the relations which require s and o should be both PER entities.
4. The number of occurrences of queries that simultaneously contain the s and o in one year’s query logs. The intuition is that the more the number of times this pair is searched, the more likely it is a true pair. The query logs we use are from Baidu (the same below).
5. The number of occurrences of queries that simultaneously contain the s and p in one year’s query logs.
6. The number of occurrences of queries that simultaneously contain the p and o in one year’s query logs.
7. Whether o is unique under the relation. The value is 1 if it is unique, otherwise 0. For example, under the relation “出生地” (‘birthplace’), o is unique, while under the relation “对友” (‘teammate’), o is not unique.
8. The similarity between $s + p$ and o. We use a similarity model developed by Baidu to compute the similarity of two queries. If the URLs that the two queries lead to in query logs are highly coincident, the two queries have a high similarity according to the model. The training data for this model is one year’s query logs from Baidu, and a neural network is used as the training model.
9. The distance between the s and the nearest negative word, the distance is regarded as 50 if there is no negative word in the sentence.
10. The distance between the o and the nearest negative word, the distance is regarded as 50 if there is no negative word in the sentence.
11. Whether there is a negative word in the sentence, 1 if it is, otherwise 0.
12. The average value of all the structural features above.

Like structural features, some semantic features can be split into 2 to 5 features. Taking the 9th feature as an example, this feature can be split into 3 features: whether the distance between the s and the nearest negative word is between 0 to 3, between 4 to 9 or more than 10. In this way, 12 features above are split into 26 features, which we call semantic features.

References

[1] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.

[2] F.M. Suchanek, G. Kasneci, G. Weikum, Yago:a core of semantic knowledge, in: International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May, 2007, pp. 697–706.

[3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka, T.M. Mitchell, Toward an architecture for never-ending language learning, in: Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010, pp. 1306–1313.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A nucleus for a web of open data, in: The Semantic Web, International Semantic Web Conference, Asian Semantic Web Conference, ISWC 2007 + Aswc 2007, Busan, Korea, November, 2007, pp. 722–735.

[5] J. Jiang, Multi-task transfer learning for weakly-supervised relation extraction, in: ACL 2009, Proceedings of the Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the Afnlp, 2-7 August 2009, Singapore, 2009.

[6] T. Evgeniou, M. Pontil, Regularized multi–task learning, in: Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, Usa, August, 2004, pp. 109–117.

[7] K. Yu, V. Tresp, A. Schwaighofer, Learning gaussian processes from multiple tasks, in: International Conference on Machine Learning, 2005, pp. 1012–1019.

[8] P. Rai, H.D. Iii, Infinite predictor subspace models for multitask learning, J. Mach. Learn. Res. 9 (2010) 613–620.

[9] B. Bakker, T. Heskes, Task clustering and gating for bayesian multitask learning, J. Mach. Learn. Res. 4 (May) (2003) 83–99.

[10] L. Jacob, J.-p. Vert, F.R. Bach, Clustered multi-task learning: A convex formulation, in: Advances in Neural Information Processing Systems, 2009, pp. 745–752.

[11] Y. Bengio, Learning Deep Architectures for AI, Now Publishers, 2009, pp. 1–127.

[12] H.L. Seal, Estimation of the probability of an event as a function of several independent variables, Biometrika 54 (1–2) (1967) 167–179.

[13] A. Rodriguez, A. Laio, Machine learning. Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492.
[14] A.E. Hoerl, R.W. Kannard, K.F. Baldwin, Ridge regression:some simulations, Comm. Statist. Simulation Comput. 4 (2) (1975) 105–123.
[15] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[16] Z. Guodong, S. Jian, Z. Jie, Z. Min, Exploring various knowledge in relation extraction, in: ACL 2005, Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA, 2005, pp. 419–444.
[17] L. Qian, G. Zhou, F. Kong, Q. Zhu, P. Qian, Exploiting constituent dependencies for tree kernel-based semantic relation extraction, in: Proceedings of the 22nd International Conference on Computational Linguistics, vol. 1, Association for Computational Linguistics, 2008, pp. 697–704.
[18] M. Pershina, B. Min, W. Xu, R. Grishman, Infusion of labeled data into distant supervision for relation extraction, in: ACL (2), 2014, pp. 732–738.
[19] M. Miwa, M. Bansal, End-to-end relation extraction using LSTMs on sequences and tree structures, in: Meeting of the Association for Computational Linguistics, pp. 1105–1116.
[20] M. Zhang, Y. Zhang, G. Fu, End-to-End neural relation extraction with global optimization, in: Conference on Empirical Methods in Natural Language Processing, 2017, pp. 1730–1740.
[21] R. Katuwal, P. Suganthan, Enhancing multi-class classification of random forest using random vector functional neural network and oblique decision surfaces, 2018, arXiv preprint arXiv:1802.01240.
[22] M. Surdeanu, J. Tibshirani, R. Nallapati, C.D. Manning, Multi-instance multi-label learning for relation extraction, in: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012, pp. 455–465.
[23] J. Jiang, C. Zhai, A systematic exploration of the feature space for relation extraction, in: HLT-NAACL, 2007, pp. 113–120.
[24] F.M. Suchanek, G. Ifrim, G. Weikum, Combining linguistic and statistical analysis to extract relations from web documents, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 712–717.
[25] M. Zhang, J. Zhang, J. Su, Exploring syntactic features for relation extraction using a convolution tree kernel, in: Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Association for Computational Linguistics, 2006, pp. 288–295.
[26] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, et al., Relation classification via convolutional deep neural network, in: COLING, 2014, pp. 2335–2344.
[27] M. Mintz, S. Bills, R. Snow, J. Dan, Distant supervision for relation extraction without labeled data, in: Joint Conference of the Meeting of the ACL and the International Joint Conference on Natural Language Processing of the Afnlp: Volume, 2009, pp. 1003–1011.
[28] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ACM, 2008, pp. 1247–1250.
[29] T.H. Nguyen, R. Grishman, Combining neural networks and log-linear models to improve relation extraction, 2015, arXiv preprint arXiv:1511.05926.
[30] P. Verga, D. Belanger, E. Strubell, B. Roth, A. McCallum, Multilingual relation extraction using compositional universal schema, 2015, arXiv preprint arXiv:1511.06396.
[31] G. Ji, K. Liu, S. He, J. Zhao, et al., Distant supervision for relation extraction with sentence-level attention and entity descriptions, in: AAAI, 2017, pp. 3060–3066.
[32] D. Zeng, K. Liu, Y. Chen, J. Zhao, Distant supervision for relation extraction via piecewise convolutional neural networks, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1753–1762.
[33] R.K. Ando, T. Zhang, A framework for learning predictive structures from multiple tasks and unlabeled data, J. Mach. Learn. Res. 6 (3) (2004) 1817–1853.
[34] H. Yang, I. King, M.R. Lyu, Online learning for multi-task feature selection, in: ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October, 2010, pp. 1693–1696.
[35] Y. Amit, M. Fink, N. Srebro, S. Ullman, Uncovering shared structures in multiclass classification, in: Machine Learning, Proceedings of the Twenty-Fourth International Conference, 2007, pp. 17–24.
[36] D. Hernández-Lobato, J.M. Hernández-Lobato, Learning feature selection dependencies in multi-task learning, Adv. Neural Inf. Process. Syst. (2013) 746–754.
[37] Y. Xue, X. Liao, L. Carin, B. Krishnapuram, Learning multiple classifiers with dirichlet process mixture priors, 2005.
[38] Q. Liu, X. Liao, L. Carin, Semi-supervised multitask learning, in: Advances in Neural Information Processing Systems, 2008, pp. 937–944.
[39] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 160–167.