Semi-automatic Knowledge Graph Construction by Relation Pattern Extraction

Yingju Xia, Zhongguang Zheng, Yao Meng, Jun Sun

Fujitsu Research and Development Center Co.,LTD
yjxia@cn.fujitsu.com

Abstract. Knowledge graphs represent information in the form of entities and relationships between them. A knowledge graph consists of multi-relational data, having entities as nodes and relations as edges. The relation indicates a relationship between these two entities. Relation extraction is the key step to construct a knowledge graph. Conventional relation extraction methods usually need large scale labeled samples for each website. It's difficult to deal with the large number of relations and the various representations of each relation. This paper proposed a novel semi-automatic method that builds knowledge graph by extracting relation patterns and finding new relations. The proposed method models the relation pattern as a tag sequence and learns the pattern similarity metric using the existing relation instances. The pattern similarity is adopted to extract new patterns for existing relations. The new relations are detected by using the pattern similarity and clustering technique. The experimental results on large scale web pages show the effectiveness and efficiency of the proposed method.

Keywords: Semantic Web, Relation Extraction, Knowledge Graph.

1 Introduction

Knowledge graphs model information in the form of entities and relationships between them [1]. This kind of relational knowledge representation has a long history in logic and artificial intelligence [2], for example, in semantic networks [3] and frames [4]. It has been used in the Semantic Web community with the purpose of creating a "web of data" that is readable by machines [5].

Knowledge graph is a powerful tool for supporting a large spectrum of search applications including ranking, recommendation, exploratory search, and web search [6, 7]. A knowledge graph aggregates information around entities across multiple content sources and links these entities together.

There is a growing interest in automatically constructing knowledge graphs [8, 9, 10]. But automatically constructing such graphs from noisy extractions presents numerous challenges [11, 12]. There are many literatures related to this topic. From the early information extraction [13, 14, 15] to special data extraction, e. g. the web table extraction [16, 17], and further, the relation extraction [18, 19, 20]. The methods range from rule based methods [21, 22] to supervised methods and semi-supervised methods [23-31].

In this study, we focus on extracting the special information from structured web and building a knowledge graph. A sample web page is shown in Fig. 1. We extract the structure information shown in the red rectangle and build a knowledge graph about the enterprises. Each page on this kind of websites contains one or more facts about a particular entity (defined as topic entity which is the subject for all relations in this page). For example, the sample web page in Fig. 1 gives information such as "Date of Establishment", "Head Office" and "Capitalization" about a company. The company entity will be the subject of all the relations and can be omitted, in this case, the relations can be represented as (relation, object). Take Fig. 1 for example, there will be some relations such as (*Date of Establishment*, "*February 6, 1936*"), (*Representative Director*, "*Yoshinori Yamashita*") and (*Capitalization*, "*135.3 billion yen*").

However, building knowledge graph from webpages is not easy. There are two main problems:

1) There are always various representations for one relation. For example, the relation "Date of Establishment" on a company website may be presented as "*Date of Establishment*", "*establishment date*", "*establishment day*", "*Date of Company Established*" and "Found date". It's hard to find all the possible descriptions.

2) There are always various templates to generate relation (relation, object) among different websites thus makes the structure or layout, differ from website to website. Take the "Required Education" of the company jobs for example, the XPath On the website (www.careerbuilder.com) is "/html/body/table/tbody/tr/td/table/tbody/tr[2]/td/table/tbody/tr[8]/td[1]". While it is "/html/body/div[1]/div[2]/div[1]/div[4]/div[1]/div/dl/dt" on another website (www.monster.com),.

Furthermore, the templates will change due to website revisions. Even in the webpages generated from the same template, the pages may differ due to the missing fields, varying numbers of instances and conditional formatting. All these problems make the relation extraction much difficult.

The conversional relation extraction methods learn extraction patterns from manual annotations [6, 24, 26, 27, 32 and 33]. The manual annotation is an expensive and time-consuming step. The CERES system [24] uses an entity-linking step in the annotation process to identify detailed page topic entities, followed by a clustering process to identify the areas of pages corresponding to each relation. This method can compete with annotation-based techniques in the literature.

This paper presents a novel semi-automatic knowledge graph construction method with relation pattern extraction using similarity learning. The knowledge graph is building from structured web page. Each web page is presented as a DOM Tree [34], the sample (relation, object) is presented as tag sequence of the XPath. The vector of the (relation, object) pair is gotten from the embedding method and feed to the Siamese network to learn a similarity metric. The relation pattern is built from the seed instance and be continually optimized by iterative steps. The knowledge graph can be built in a semi-automatic way. Given some instances of the relations for an entity, the system build the relation patterns and find more relation instances by the similarity metric. The new relation instances are also used to refine the existing relation patterns. The system can also find new relations by clustering method using the learned similarity metric. For example, to build a knowledge graph for enterprise, the system only need some instances for the existing relations ("Date of Establishment", "Capitalization", "address", "website" and so on), the system builds pattern for each relation and extracts information from web pages. The system find more relation instances from web pages and refine the relation patterns. By using the similarity metric learnt from relation instances and the clustering approach, the system can find new relations such as "slogan" and build pattern for it.

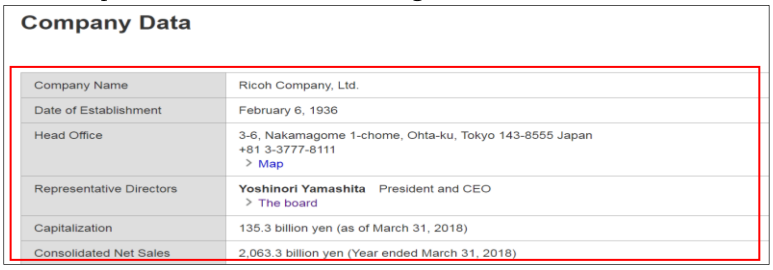
The main contributions of this paper are listed as following:

1) A method is proposed that using tag sequence and it's embedding to build the relation patterns.

2) The relation extraction pattern similarity is learnt from the tag sequence of seed instances by using a Siamese network. The relation patterns can achieve self-improvement by finding more and more instances using the similarity metric.

3) The proposed method can also be used to detect the new relations and build the extraction patterns for the new relations.

The rest of this paper is organized as follows. Section 2 presents the knowledge graph building method using pattern similarity based relation extraction. Section 3 shows the experimental results. Section 4 gives several conclusions and future works.



Company Data

Company Name	Ricoh Company, Ltd.
Date of Establishment	February 6, 1936
Head Office	3-6, Nakamagome 1-chome, Ohta-ku, Tokyo 143-8555 Japan +81 3-3777-8111 > Map
Representative Directors	**Yoshinori Yamashita** President and CEO > The board
Capitalization	135.3 billion yen (as of March 31, 2018)
Consolidated Net Sales	2,063.3 billion yen (Year ended March 31, 2018)

Fig. 1. A sample web page about a company.

2 Method Introduction

There are several steps in our system:
1) Make the representation for the relation instances.
2) Learn the similarity between tag sequences.
3) Build extraction patterns from seed instances.
4) Refine the extraction patterns with new instances.
Following are the details for these steps.

2.1 Representation of the Input Relation Instances

The inputs of this system are two relation instances from the webpages. The *relation* (*R*) and *object* (*O*) of each relation instance (*R*, O) will be embedded in a tag sequence of XPath like this:

<tag_{R1}> <tag_{R2}> … … <tag_{Rm}> *R* <tag_{O1}> <tag_{O2}> … <tag_{On}> *O*

This tag sequence will be used to present the relation instance.

Take the "*Capitalization*" relation as example, the relation instance is (*Capitalization*, *135.3 billion yen*) and the XPath for these two nodes are:
/html/body/table/tbody/tr/td/table/tbody/tr[2]/td/table/tbody/tr[8]/td[1] Capitalization
/html/body/table/tbody/tr/td/table/tbody/tr[2]/td/table/tbody/tr[8]/td[2] 135.3 billion yen

This relation instance is represented as a tag sequence:

(*<html> <body> <table> <tbody> <tr> <td> <table> <tbody> <tr[2]> <td> <table> <tbody> <tr[8]> <td[1]> Capitalization <html> <body> <table> <tbody> <tr> <td> <table> <tbody> <tr[2]> <td> <table> <tbody> <tr[8]> <td[2]> 135.3 billion yen*)

This tag sequence presents the layout information on the web page.

The idea of this paper is to learn the similarity between relation instances and build the relation pattern using the similarity. It is hard to give the similarity of the relation instances pair, but it is easy to give the label that whether these two relation instances belong to the same relation or not. In our system, we use '0' to indicate the same relation and '1' for different relations.

For example, if we have another relation instance (*capital fund*, *$202.5 billion*) and the tag sequence:

(*<html> <body> <div[1]> <div[2]> <div[1]> <div[4] > <div[1]> <div> <dl> <dt[6] capital fund <html> <body> <div[1]> <div[2]> <div[1]> <div[4]> <div[1]> <div> <dl> <dd[7]> $202.5 billion*)

When we put these two tag sequences to the system, we also should give the label '0' (same relation). It is a training instance for the system.

2.2 The Siamese Network for Sequence Similarity Calculation

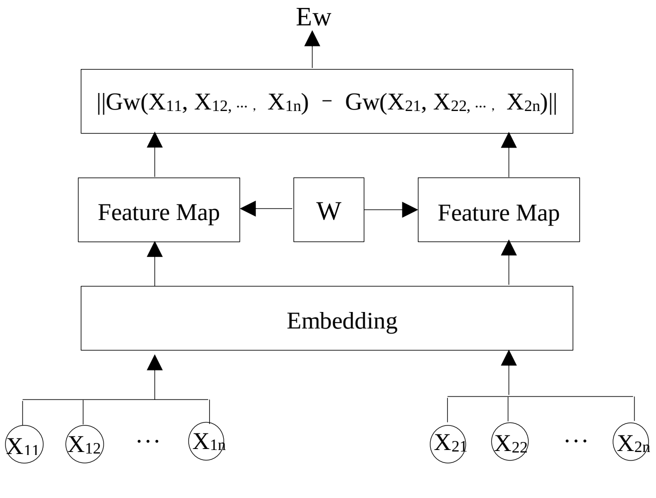


Fig. 2. Similarity learning with Siamese Network.

The sequence similarity learning is the key step in our system. Fig. 2 shows our similarity learning method.

The first step of this method is the tag sequence embedding. That is to make a vector for the input tag sequence so that similar tag sequences or tag sequence used in a similar context are close to each other in the vector space. In the free text analysis field, the word embedding is widely used, particularly in deep learning applications. The word embeddings are a set of feature engineering techniques that transform sparse vector representations of words into a dense, continuous vector space, enabling system to identify similarities between words and phrases based on their context.

In this paper, we adopt the word embedding approach [35, 36] and trained a Skip-Gram word2vec model from the intermediate result of DOM tree parsing. In Fig. 2, the X_{11}, X_{12}, …, and X_{1n} are tags for the first tag sequence and the X_{21}, X_{22}, …, and X_{2n} for the second tag sequence. They will be convert into vectors through the embedding component. After that, these tag embeddings are combined into one vector as the embedding of the tag sequence. There are several ways to combine the tag embedings. In this paper, we chose the concatenation operation due to experimental results. The concatenation operation is to concatenate the vectors of each tag one by one to make a big vector. Say, if we have 10 tag vectors and each vector has the dimension 128, then the concatenation vector will has the dimension 1280.

The vectors of the tag sequence are put into the feature map layers. The feature maps layer converts the tag sequence into a target space such that a simple distance in the target space approximates the “semantic” distance in the input space. Since the two feature maps layer share the same parameter *W*, this can be consider as the Siamese architecture [37, 38]. This network is suitable for recognition or verification applications where the number of categories is very large and not known during training.

Given a family of functions $Gw(x)$ parameterized by *W*, the method seeks to find a *W* such that the similarity metric $Ew(X_1, X_2) = \| Gw(X_1) - Gw(X_2) \|$ is small if X_1 and X_2 belong to the same extraction pattern, and large if they belong to different patterns. In our system, the structure of the feature map network is a 5 layers full-connected network. The output dimension of the feature map is set to 128.

In the training stage, let *Y* be a binary label of the pair, *Y*=0 if the X_1 and X_2 belong to the same relation (genuine pair) and $Y=1$ otherwise (impostor pair). Let *W* be the shared parameter vector that is subject to learning, and let $Gw(X_1)$ and $Gw(X_2)$ be the two points in the low-dimensional space that are generated by mapping and X_1 and X_2. Then our system use the

$Ew(X_1, X_2)$ to measures the similarity between X_1 and X_2.

It is defined as $Ew(X_1, X_2) = \| Gw(X_1) - Gw(X_2) \|$

The loss function is of the form:

$$\ell(W) = \sum_{i=1}^{p} L(W, (Y, X_1, X_2)^i) \tag{1}$$

$$L(W, (Y, X_1, X_2)^i) = (1 - Y)L_G(Ew(X_1, X_2)^i) + YL_I(Ew(X_1, X_2)^i) \tag{2}$$

Where $(Y, X_1, X_2)^i$ is the i-th sample, which is composed of a pair of samples and a label, L_G is the partial loss function for a genuine pair, L_I the partial loss function for an impostor pair, and P the number of training samples. L_G and L_I should be designed in such a way that minimization of L will decrease the energy of genuine pairs and increase the energy of impostor pairs. A simple way to achieve that is to make L_G monotonically increasing, and L_I monotonically decreasing.

In this paper, the L_G and L_I are:

$$L_G = \frac{2}{Q}(Ew)^3 \tag{3}$$

$$L_I = Qe^{-\frac{Ew}{Q}} \tag{4}$$

Here the Q is a constant and is set to the upper bound of Ew

The Ew is the similarity metric which learned by the Siamese network, it is in the range [0, 1].

2.3 Relation Patterns building and refining

Once we have learnt the similarity metric, we can use it to calculate the similarity of two input tag sequences. The tag sequence pair with similarity bigger than a given threshold can be used to build the same relation pattern. That means that, if we have some seed instances, we can use them and the similarity metric to find more similar instances. And build the relation patterns from these instances.

How to build the relation pattern using the instances? There are several ways to do this. In the rule scenario, we can deduce the regular expression from the detected relation instances and use it as the relation pattern. In this paper, we use the centroid point as the relation pattern.

The key technique of this system is the similarity learning method. With the similarity metric, we can collect more and more relation instances and then build better extraction pattern. Iteratively, the extraction pattern is used to find more relation instances. The experimental results in Session 3 shows the performance of our similarity learning method. For example, in the "*Capitalization*" scenario, we have the instance:

(*<html> <body> <table> <tbody> <tr> <td> <table> <tbody> <tr[2]> <td> <table> <tbody> <tr[8]> <td[1]> Capitalization <html> <body> <table> <tbody> <tr> <td> <table> <tbody> <tr[2]> <td> <table> <tbody> <tr[8]> <td[2]> 135.3 billion yen*)

and

(*<html> <body> <div[1]> <div[2]> <div[1]> <div[4] > <div[1]> <div> <dl> <dt[6] capital fund <html> <body> <div[1]> <div[2]> <div[1]> <div[4]> <div[1]> <div> <dl> <dd[7]> $202.5 billion*)

After some iterations, we find some new tag sequence belong to the "*Capitalization*" relation, say,

(*<html> <body> <div[1]> <div[3]> <div[2]> <div> <div[7]> <div[1]> capital amount <html> <body> <div[1]> <div[3]> <div[2]> <div> <div[7]> <div[2] US$65,000,000*)

These new tag sequences are put into the collection of the "*Capitalization*" relation and used to update the relation pattern. Then the updated relation pattern is used to collect new relation instances.

This method can also be used to detect the patterns for new relations. For example, if we already have some relations about the job such as the "*Date of Establishment*", "*Capitalization*", "*address*" and "*website*". The proposed system may get some relation instances clusters using clustering method. And there may be cluster for a new relation, say, "*Number of Employees*". Here are the relation instances for this new relation:

(*<html> <body> <div[1]> <div> <div> <div[2]> <div[2]> <div[1]> <div[1]> <dl[4]> <dt> Number of Employees <html> <body> <div[1]> <div> <div> <div[2]> <div[2]> <div[1]> <div[1]> <dl[4]> <dd> 98,868*)

(*<html> <body> <div[2]> <div> <dl[10]> <dt> Staff number <html> <body> <div[2]> <div> <dl[10]> <dd> 1,678*)

The relation name is different ("*Number of Employees*" and "*Staff number*") and the format of the object is also different. They should be put into one cluster using the clustering method.

3 Experiments

3.1 Data set

We built a knowledge graph for enterprise using the proposed method. Firstly, we got a collection of websites about Japanese companies by search engine. Start from some manually labeled instances, we built a knowledge graph which contains 2,717,653 company entities and 22,257,276 triples.

To show the performance of our method. We conducted a set of experiments which use part of the data. We selected 10 relations to train the pattern similarity learning model. The instance number of each relation are shown in the Table 1.

Table 1. The Enterprise Knowledge Graph dataset.

Relation	Number of instance	Relation	Number of instance
Name	43,680	rel-org	15,880
Address	100,840	finance	15,218
Person	24,297	size	9,462
homepage	17,651	date	14,872
phone number	20,783	product	23,590

3.2 Experimental results

The evaluation metric for similarity are:

FA (False Accept Rate, the percentage of impostor pairs accepted)

FR (False Reject Rate, the percentage of genuine pairs rejected)

EER (Equal Error Rate, the point where FA equals FR).

To train and evaluate the similarity module, we need to build a set of tag sequence pairs, including genuine pairs and impostor pairs. There are 10 relations and total 286,276 instances. If we random select 1000 instances from each relation, we can build 9,999,000 genuine pairs and 90,000,000 impostor pairs.

Table 2 shows the experimental results of similarity learning, the system get EER 0.01 using the 10,000 instances (1,000 instances per relation). The iteration steps is about 500 to get the performance. The Fig. 3 shows the EER at different instance number. We can see that the more instances, the better performance. It trends to convergence when the instances number of each relation is about 1000.

Table 2. Experimental Results of Similarity Learning

Instance number	Iteration steps	EER
10,000	100	0.04
10,000	500	0.01

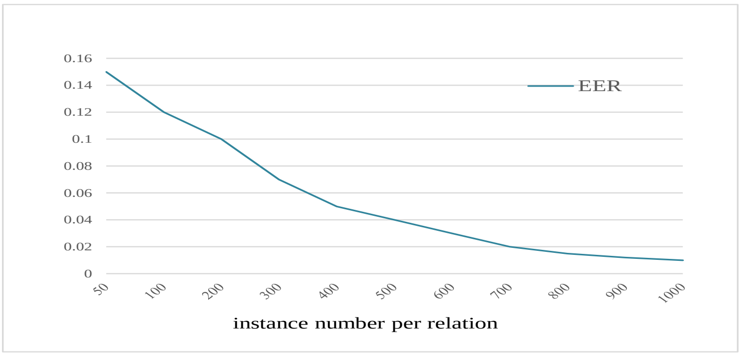


Fig. 3. The impact of the instance number.

The main scenario for the proposed method is to find new patterns and new relations. To evaluate the performance, we conduct experiments on unseen dataset. The unseen dataset means the test data are separated from the training data. In the enterprise knowledge graph case, we train the model on the some relations and test on the dataset with other relations. More concretely, we train the model on the previous 10 relations (name, address, person, homepage, phone number, rel-org, finance, size, date, and product). Then we test this model on the dataset with different relations (say, fax, email, Permission, Introduction, domain, office) shown on table 3. We find the optimal threshold on the valid dataset and use the threshold to get the FA and FR on test dataset. The EER is gotten from test dataset. The system got EER 0.05 on the unseen datasets. This enable the new patterns and relations find procedure.

Table 3. The unseen dataset.

Relation	**Number of instance**	**Relation**	**Number of instance**
fax	5,911	introduction	4,436
email	3,586	domains	13,347
permission	2,645	offices	11,145

In the scenarios that finding the new relations and getting the patterns, the clustering approach is adopted to put the similar instances into same group.

To evaluate the new relation detection performance. We adopt the following steps and use the precision and recall metrics.

1) Input test data of k (for example, k=6) categories
2) Clustering, output k clusters
3) Assign label for data in each cluster
4) Compare the assigned labels with the true labels
5) Using precision and recall (by Table 4) to evaluate the performance
   Precision = TP/(TP+FP)
   Recall = TP /(TP+FN)

Two clustering methods (Hierarchy and K-Means) are evaluated. The experiments results shown in Table 5. From this table, we can see that K-Means got better results than hierarchy clustering method. It can be used to find the new relations.

Table 4. The metric for evaluated the clustering results

		Clustering labels	
		Y	N
True	P	True Positives (TP)	False Negatives (FN)
Label	N	False Positives (FP)	True Negatives (TN)

Table 5. The clustering experimental results

	Hierarchy clustering		**K-Means clustering**	
	precision	recall	precision	recall
fax	0.6976	0.775	0.7284	0.912
email	0.7847	0.758	0.9134	0.77
permission	0.7313	0.675	0.7414	0.671
introduction	0.9973	0.366	0.9989	0.882
domains	0.5903	1.00	0.8503	1.00
offices	0.9915	0.931	0.9883	0.93

We also evaluated our method on open dataset SWDE [26]. The SWDE dataset consists about 124K pages collected from 80 websites. These websites are related to 8 semantically diverse verticals, including *Autos, Books, Cameras, Jobs, Movies, NBA Players, Restaurants and Universities*. Table 6 give the overview of SWDE dataset.

Table 6. The Overview of SWDE dataset.

Vertical	#Sites	#Pages	Relations
Autos	10	17,923	model, price, engine, fuel-economy
Book	10	20,000	title, author, ISBN-13, publisher, publication_date
Cameras	10	5,258	model, price, manufacturer
Jobs	10	20,000	title, company, location, date
Movie	10	20,000	title, director, genre, rating
NBA Player	10	4,405	name, height, team, weight
Restaurants	10	20,000	name, address, phone, cuisine
University	10	16,705	name, phone, website, type

We also use 1000 instances for each relation. The similarity learning results are shown in Table 7. The best results is got from ‘Restaurants’ which is close to our enterprise dataset. There are some verticals such as ‘Book’ and ‘Movie’ got lower performance due to the complex of the websites.

Table 7. Similarity Evaluation Results on SWDE dataset

Vertical	EER	Vertical	EER
Autos	0.050	Movie	0.102
Book	0.120	NBA Player	0.072
Cameras	0.082	Restaurants	0.010
Jobs	0.118	University	0.065

4 Conclusion

This paper presents a new semi-automatic knowledge graph construction method by relation pattern extraction. The proposed method uses tag sequence to build the relation pattern. A Siamese network is adopted to learn the similarity metric from the tag sequences. The proposed method builds the relation patterns and continually refines those patterns by finding more and more samples using the similarity metric. It can also be used to detect patterns for the new relations.

The future work includes finding new representation of the relation patterns and give the labels for new relations automatically.

References

1. Nickel M, Murphy K, Tresp V, et al. A review of relational machine learning for knowledge graphs[J]. Proceedings of the IEEE, 2016, 104(1): 11-33.

2. Davis R, Shrobe H, and Szolovits P. What is a knowledge representation? AI Magazine, 1993, vol. 14, no. 1, 17–33.
3. Sowa J. Semantic networks. Encyclopedia of Cognitive Science, 2006.
4. Minsky M. A framework for representing knowledge. MIT-AI Laboratory Memo 306, 1974.
5. Berners-Lee T, Hendler J, and Lassila O. The Semantic Web, 2001. http://www.scientificamerican.com/article/the-semantic-web/
6. Dong X, Gabrilovich E, Heitz G, et al. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014: 601-610.
7. Voskarides N, Meij E, Tsagkias M, et al. Learning to explain entity relationships in knowledge graphs. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. 2015, 1: 564-574.
8. Fan J, Hoffman R, Kalyanpur A, et al. Automatic knowledge extraction from documents. Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction. 2012.
9. Craven M, DiPasquo D, Freitag D, et al. Learning to construct knowledge bases from the World Wide Web[J]. Artificial intelligence, 2000, 118(1-2): 69-113.
10. Weston J, Bordes A, Yakhnenko O, et al. Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction. Conference on Empirical Methods in Natural Language Processing. 2013: 1366-1371.
11. Jiang S, Lowd D, Dou D. Learning to refine an automatically extracted knowledge base using markov logic. IEEE 12th International Conference on Data Mining (ICDM), 2012: 912-917.
12. Bordes A, Glorot X, Weston J, et al. A semantic matching energy function for learning with multi-relational data. Machine Learning, 2014, 94(2): 233-259.
13. Ferrara E, De Meo P, Fiumara G, et al. Web data extraction, applications and techniques: A survey. Knowledge-based systems, 2014, 70: 301-323.
14. Gottlob G, Koch C, Pieris A. Logic, Languages, and Rules for Web Data Extraction and Reasoning over Data. International Conference on Language and Automata Theory and Applications. Springer, Cham, 2017: 27-47.
15. Yao X, Van Durme B. Information extraction over structured data: Question answering with freebase. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2014, 1: 956-966.
16. Lehmberg O, Ritze D, Meusel R, et al. A large public corpus of web tables containing time and context metadata. Proceedings of the 25th International Conference Companion on World Wide Web. 2016: 75-76.
17. Ritze D, Lehmberg O, Oulabi Y, et al. Profiling the potential of web tables for augmenting cross-domain knowledge bases. Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2016: 251-261.
18. Song M, Kim W C, Lee D, et al. PKDE4J: Entity and relation extraction for public knowledge discovery. Journal of biomedical informatics, 2015, 57: 320-332.
19. Ji G, Liu K, He S, et al. Distant Supervision for Relation Extraction with Sentence-Level Attention and Entity Descriptions. AAAI. 2017: 3060-3066.
20. Heist N, Paulheim H. Language-agnostic relation extraction from wikipedia abstracts. International Semantic Web Conference. Springer, Cham, 2017: 383-399.

21. Soderland S. Learning information extraction rules for semi-structured and free text. Machine learning, 1999, 34(1-3): 233-272.
22. Chang C H, Kayed M, Girgis M R, et al. A survey of web information extraction systems. IEEE transactions on knowledge and data engineering, 2006, 18(10): 1411-1428.
23. Mintz M, Bills S, Snow R, et al. Distant supervision for relation extraction without labeled data. Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2. Association for Computational Linguistics, 2009: 1003-1011.
24. Lockard, C., Dong, X. L., Einolghozati A, et al.: CERES: Distantly Supervised Relation Extraction from the Semi-Structured Web. arXiv:1804.04635, 2018..
25. Hoffmann R, Zhang C, Ling X, et al. Knowledge-based weak supervision for information extraction of overlapping relations. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011: 541-550.
26. Hao, Q., Cai, R., Pang, Y. et al.: From one tree to a forest: a unified solution for structured web data extraction. Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 2011: 775-784..
27. Gentile, A. L., Zhang, Z., Ciravegna, F.: Early steps towards web scale information extraction with lodie. AI Magazine, 36:55–64, 2015.
28. Cohen, J. P., Ding, W., Bagherjeiran, A.: Semi-supervised web wrapper repair via recursive tree matching. CoRR,abs/1505.01303, 2015.
29. Mironczuk, M.: The biggrams: the semi-supervised information extraction system from html: an improvement in the wrapper induction. Knowledge and Information Systems, pages 1–66, 2017.
30. Ortona, S., Orsi, G., Buoncristiano, M., Furche, T.: Wadar: Joint wrapper and data re-pair. PVLDB, 8:1996–1999, 2015.
31. Bronzi, M., Crescenzi, V., Merialdo, P., Papotti, P.: Extraction and integration of partially overlapping web sources. PVLDB, 6:805–816, 2013.
32. Kushmerick, N., Weld, D. S., Doorenbos, R.: Wrapper induction for information extraction. In IJCAI, 1997.
33. Gulhane, P., Madaan, A., Mehta, R. R.: Ramamirtham,R. Rastogi, S. Satpal, S. H. Sengamedu, A. Tengli, andC. Tiwari. Web-scale information extraction with Vertex. The 27th International Conference on Data Engineering, pages 1209–1220, 2011.
34. 34. W3C, XML Path Language (XPath) Version 1.0, W3C Recommendation, 1999, http://www.w3.org/TR/xpath.
35. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. Advances in neural information processing systems. 2014: 2177-2185.
36. Mikolov, T.,Sutskever, I., Chen, K.: Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems. 2013: 3111-3119
37. Bromley, J., Guyon, I., LeCun, Y.: Signature verification using a "siamese" time delay neural network. Advances in Neural Information Processing Systems. 1994: 737-744.
38. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. Computer Vision and Pattern Recognition, 2005., 1: 539-546.