Web Data Extraction with Seed Samples

Yingju Xia, Jun Ma, Zhongguang Zheng, Rujie Liu

Fujitsu Research and Development Center Co.,LTD
355Unit 3F, Gate 6, Space 8,Pacific Century Place, No.2A Gong Ti Bei Lu, Chaoyang District, Beijing, China
yjxia@cn.fujitsu.com

Abstract. The web which contains a large scale of semi-structured data has been a rich source for populating knowledge databases. Much effort has been attracted to extract structured data from enormous websites. This paper proposes a new method that extracts web data from seed samples. The proposed method learns a similarity metric using a Siamese network with the seed samples. The extraction patterns are built from the seed samples and are continuously optimized by finding more and more samples using the similarity metric. The experiments on large scale web pages show the effectiveness and efficiency of the proposed method.
Keywords: semi-structured data mining, extraction pattern, similarity learning.

1 Introduction

The abundance of web data has made web data extraction as an essential tool in data acquisition. There are many literatures related to extract information from Web [1]. Among these web data extraction tools, the relation extraction [2, 3, 4] aims to extract the triples (*subject*, *relation*, *object*) to provide factual information. For example, the triple (*The Old Man and the Sea*, *author*, *Ernest Hemingway*) descripts the fact that the author of the short novel *"The Old Man and the Sea"* is *Ernest Hemingway*. From the perspective of the data source, the relation extraction has two main categories: from free text and from the structured web pages. The techniques for extracting relation from free text are main natural language processing techniques such as part of speech tagging, parsing, lexical analysis, syntactic analysis, and semantic analysis. While for the structured web data source, the techniques can be traced back to the wrapper induction. Given a website, wrapper induction asks for manual annotations, often on a handful of pages, and derives the extraction patterns (usually presented as XPaths[5, 6]) that can be applied to the whole website.

In this study, we focus on extracting the relations from semi-structured websites which are the most promising knowledge sources. This kind of websites are typically populated by data from large databases and thus the pages from the same template sharing a common structure of information [7]. Each page 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, in the Internet Movie Database (IMDb www.imdb.com),

each page gives information such as title, director, writer and stars of a particular film. The film will be the *subject* of all the relations and can be omitted, in this case, the relations can be represented as (*relation*, *object*). For example, on the page about film "DIE HARD", there will be some relations such as *(Director*, *John McTiernan*), (*Writer*, *Roderick Thorp*) and (*Star*, *Bruce Willis*).

However, relation extraction from webpages is not easy. There are two main problems:

1) There are always various representations for one relation. For example, the relation "*education*" on a Job website may be presented as "*Education*", "*Required Education*", "*Education Level*" and "*Required Education Level*". It's hard to find all the possible descriptions.

2) There are always various templates to generate the triples (*subject, relation, object*) among different websites thus makes the structure or layout, differ from website to website. Take the XPath of "*Required Education*" 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 on the other website (www.monster.com), it is "*/html/body/div[1]/div[2]/div[1]/div[4]/div[1]/div/dl/dt[6] ".*

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 [8, 9, 10, 11 and 12]. The manual annotation is an expensive and time-consuming step. The CERES system [7] 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 in terms of extraction quality.

This paper presents a new relation extraction method with seed samples. Each page is presented as a DOM Tree, 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 extraction pattern is built from the seed samples and be continually optimized by iterative steps.

The main contributions of this paper are listed as following:

1) A method is proposed that using tag sequence of the XPath and its embedding to build the relation extraction patterns.

2) The relation extraction pattern similarity is learnt from the tag sequence of seed samples by using a Siamese network. The relation extraction patterns can achieve self-improvement by finding more and more samples 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 proposed method. Section 3 shows the experimental results. Section 4 gives several conclusions and future works.

2 Method Introduction

There are several steps in our system:

1) Make the representation for the relation samples
2) Learn the similarity between tag sequences
3) Build extraction patterns from seed samples
4) Refine the extraction patterns with iterative steps

Following are the details for these steps.

2.1 Representation of the Input Relation Samples

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

$<tag_{R1}> <tag_{R2}> \ldots \ldots <tag_{Rm}>$ *R* $<tag_{O1}> <tag_{O2}> \ldots <tag_{On}>$ *O*

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

Take the "*Required Education*" relation as example, the relation instance is (*Required Education, 4 year degree*) 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] Required Education

/html/body/table/tbody/tr/td/table/tbody/tr[2]/td/table/tbody/tr[8]/td[2] 4 Year Degree

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]> Required Education <html> <body> <table> <tbody> <tr> <td> <table> <tbody> <tr[2]> <td> <table> <tbody> <tr[8]> <td[2]> 4 Year Degree*)

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 extraction 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 (*Education Level*, *Bachelor's Degree*) and the tag sequence:

(*<html> <body> <div[1]> <div[2]> <div[1]> <div[4] > <div[1]> <dl> <dt[6] Education Level <html> <body> <div[1]> <div[2]> <div[1]> <div[4]> <div[1]> <dl> <dd[7]> Bachelor's Degree*)

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

2.2 The Siamese Network for Sequence Similarity Calculation

The sequence similarity calculation is the key step in our system. Fig. 1 shows our similarity learning method.

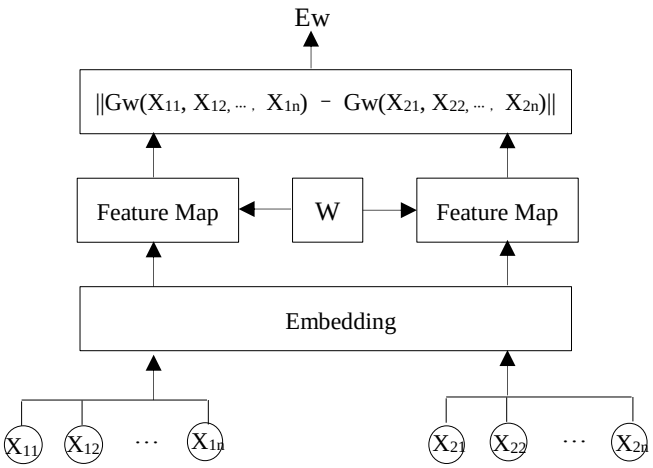

Fig. 1. Similarity learning with Siamese Network

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 on a large scale based on their context.

In this paper, we adopt the word embedding approach [13, 14] and trained a Skip-Gram word2vec model from the intermediate result of DOM tree parsing. In Fig. 1, 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 converted 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 embeddings. 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 have 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 [15, 16]. 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 Extraction Patterns building and refining

Once we have trained the similarity calculation system, 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 extraction pattern for the given relation. That means that, if we have some seed samples, we can use them and the similarity metric to find more similar samples. And build the relation extraction patterns from these samples.

How to build the extraction pattern using the samples? 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 extraction pattern. In the vector scenario, we can use the centroid point as the extraction pattern.

In this paper, we adopt the neural network to learn a similarity metric. We cannot calculate the centroid in terms of physical interpretation. So we pick several points as the representative points of the extraction pattern. These representative points are the points which apart from each other and near to the rest points.

Once we get the representative points (or pseudo centroid points), we can use these points to classify an input tag sequence. If the similarity between the input tag sequence and these representative points is higher than the threshold, we can predict the input sequence as the same relation.

From this point of view, the extraction pattern is built from the representative points and the similarity metric. The initial extraction pattern can be built from the seed samples. The representative points can be updated with more samples finding by the similarity calculation module. The whole procedure is shown below:

Let there are *K* relations: $\{R_1, R_2, \ldots, R_k\}$ and each relation has a cluster of samples. So there are *K* clusters $\{C_1, C_2, \ldots, C_k\}$. There is also a cluster of representative points for each relation. Say, for the relation R_i, there is a P_i whose members $\{P_{i1}, P_{i2}, \ldots, P_{in}\}$ are the representative points of the R_i.

Given predefined parameter *M* (the number of representative points) or the threshold θ for representative point selection. We use the steps below to update the representative points.

1) Randomly select *n* samples from the dataset
2) For each samples, put it into the nearest cluster C_i
3) For each C_m, use the formula below to update representative points

$$\underset{i}{argmax} \sum_{\substack{j \neq i \\ Xmi \in Cm, Xmj \in Cm}} sim(x_{mi}, x_{mj}) \Big/ \sum_{\substack{k \neq m \\ Xmi \in Cm}} sim(x_{mi}, P_k) \tag{3}$$

$$sim(x_{mi}, P_k) = \sum_{\substack{j \\ X_{kj} \in P_k}} sim(x_{mi}, x_{kj}) \tag{4}$$

Select *M* samples or all the samples which similarity greater than the give threshold θ

4) Repeat 1) ~3) until convergence or reach the maximum iteration steps.

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 "*Required Education*" scenario, we have the seed sample:

(*<html> <body> <table> <tbody> <tr> <td> <table> <tbody> <tr[2]> <td> <table> <tbody> <tr[8]> <td[1]> Required Education <html> <body> <table> <tbody> <tr> <td> <table> <tbody> <tr[2]> <td> <table> <tbody> <tr[8]> <td[2]> 4 Year Degree*)

and

(*<html> <body> <div[1]> <div[2]> <div[1]> <div[4] > <div[1]> <dl> <dt[6] Education Level <html> <body> <div[1]> <div[2]> <div[1]> <div[4] <div[1]> <dl> <dd[7]> Bachelor's Degree*)

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

(*<html> <body> <div[1]> <div[3]> <div[2]> <div[7]> <div[1]> Required Education Level <html> <body> <div[1]> <div[3]> <div[2]> <div[7]> <div[2] Ph. D*)

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

This method can also be used to detect the extracting patterns for new relations. For example, if we already have some relations about the job such as the "*Job Type*", "*Employee Type*", "*Required Education*", "*Required Experience*", "*Location*" and "*Description*". The proposed system may get some relation instances clusters using clustering method. And there may be cluster for a new relation, say, "*Post Date*". Here are the relation instances for this new relation:

(*<html> <body> <div[1]>   <div[2]> <div[2]> <div[1]> <div[1]> <dl[4]> <dt> Date <html> <body> <div[1]>   <div[2]> <div[2]> <div[1]> <div[1]> <dl[4]> <dd> 11-13-2011*)

(*<html> <body> <div[2]> <dl[10]> <dt> Post Date <html> <body> <div[2]> <dl[10]> <dd> 2010-06-25*)

The relation name is different ("*Post Date*" and "*Date*") and the format of the object is also different.

3 Experiments

We conducted several experiments to evaluate our system. The dataset and experimental results are shown following.

3.1 Data Set

We employed the dataset of SWDE [10] which is an open Web Data Extraction dataset. Table 1 shows the dataset overview. The SWDE 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*. Here, the verticals refer to *topic entities* which defined in Section 1.

For each vertical, 10 popular websites were identified by issuing queries to search engines and mining the search results. For each website, 200~2000 pages were collected, each containing structured data of one entity. For each vertical, a set of (3~5) common relations were selected as the targets of data extraction.

Table 1. 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

3.2 Similarity Experimental Results

The input of the similarity calculation module is a pair of tag sequence vectors. The output of it should be small if the input tag sequence pair belong to the same relation, and large if they belong to different categories. To train and evaluate the similarity module, we need to build a set of tag sequence pairs, including genuine pairs and impostor pairs. Take the vertical "*Autos*" for example, there are 4 relations and total 17,923 pages. If we random select 1000 instances for each relation, we can build 3,996,000 genuine pairs and 12,000,000 pairs.

The Table 2 shows the genuine and impostor pairs generated for each vertical. Here the "*All verticals*" means the relations in all 8 verticals are put into one set.

Table 2. The Genuine and Impostor Pairs for each Vertical

Vertical	#genuine	#impostor	Vertical	#genuine	#impostor
Autos	3,996,000	12,000,000	Movie	3,996,000	12,000,000
Book	4,995,000	20,000,000	NBA Player	3,236,400	9,720,000
Cameras	2,997,000	6,000,000	Restaurants	3,996,000	12,000,000
Jobs	3,996,000	12,000,000	University	3,996,000	12,000,000
All verticals	31,968,000	992,000,000			

The evaluation metrics adopted here is EER (Equal Error Rate), The EER is calculated by FA (False Accept Rate) and FR (False Reject Rate). These two types of errors (FA and FR) are related to the similarity threshold. When the threshold is set higher, false acceptance errors will be reduced, but the probability of a false rejection (FR) will be higher. The EER is the point where FA equals FR. It's a tradeoff between the FA and FR and can be used as a single metric to compare two systems.

We used the data shown in Table 2 and conducted 10-fold cross validation test. The experimental results are shown in Table 3. From this table, we can see that the system get the highest performance on the vertical "*Restaurants*" which has the EER 0.01. The vertical "*Book*" performances poor due to the format varies a lot. Most of the verticals have the EER lower than 0.1, it is a good performance. The "*All verti-*

cals" category got the EER 0.078, which is also well consider the various presentations in this category.

Since each tag sequence is converted into a vector, we have conducted the experiments to evaluate the dimension of the tag and the length of sequence cross the verticals. Experimental results shown that the dimension of tag has little impact. The dimension 100, 128, 200 has no significant difference on the performance in same vertical and cross verticals. In this set of experiments, we set the dimension as 128.

However, the length of the tag sequence has larger impact on the performance than the tag dimension. Because, the input of the feature map needs fix dimension, we try several length settings on the verticals and find optimal one for each relations. In the experiments, different sequence length are setting for different relations, it is a kind of hyper-parameter. Most of the tag sequence length are set to 12, the tag sequence which length is bigger than 12 will be truncated.

Table 3. Similarity Evaluation Results

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
All verticals	0.078		

3.3 Relation Extraction Experimental Results

We adopted the conventional metric *F1* score to evaluate the relation extraction performance. The *F1* is gotten from precision and recall as:

*F1 = precision*recall*2* / (*precision* + *recall*)

We have compared our system with the state-of-the-art results on SWDE in the literatures. In these systems, Hao et al. [10], XTPath [17], BigGrams[18], LODIE-Ideal(LODIE-I)[12], RR+WADaR(RR+WAD) [19], Bronzi et al. [20] have reported the experimental results on the whole verticals of the SEDE. While, LODIE-LOD(LODIE-L)[20] and CERES [7] only report experimental results of part of the verticals.

Table 4 shows the experimental results, in all verticals, our system's performance is high than the average. In "*Restaurants*" vertical, our system got the highest performance. This is consistent with the similarity evaluation results.

The experimental results are also shown with candle chart in Fig. 2. Here the solid rectangle means that the system is higher than the average. The upper bound of the solid rectangle is the system while the lower bound of the solid rectangle is the average. If the system is lower than the average, it will be shown in a hollow rectangle. There are no hollow rectangles in the Fig. 2 since our system beat all the average on all verticals.

Although, we only got highest performance on one vertical (*Restaurants*), our system is rather stable on all verticals. Some systems got many highest performance, but

got low performance on some verticals. For example, the BigGrams system got highest performance on 4 verticals and its performance on vertical "*University*" is lower than the average. From this point of view, the XTPath system is more stable and better than our system. Fig. 3 shows the candle chart of the XTPath. Compared with our system (Fig. 2), in most verticals, it's more close to the highest performance.

Table 4. Relation Extraction Experimental Results

Vertical	Movie	NBA Player	University	Book	Auto	Job	Camera	Restaurants
Hao et al.	0.79	0.82	0.83	0.86	0.72	0.87	0.96	0.98
XTPath	0.94	0.98	0.98	0.97	0.97	0.96	0.95	0.97
BigGrams	0.90	0.9	0.79	0.99	0.99	0.98	0.96	0.98
LODIE-I	0.86	0.9	0.96	0.85	0.94	0.82	0.76	0.89
LODIE-L	0.76	0.87	0.91	0.78	NA	NA	NA	0.69
RR+WAD	0.73	0.8	0.79	0.7	0.85	0.74	0.73	0.83
Bronzi et al.	0.93	0.89	0.97	0.91	0.9	0.49	0.86	0.98
CERES	0.99	0.98	0.94	0.76	NA	NA	NA	NA
Our System	0.89	0.92	0.93	0.88	0.95	0.9	0.91	0.98

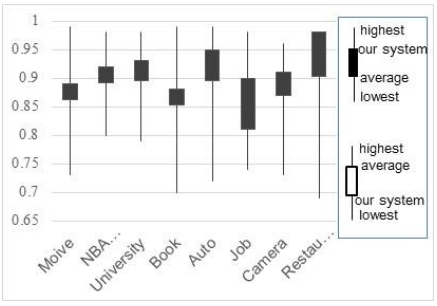

Fig. 2. The candle chart of our system

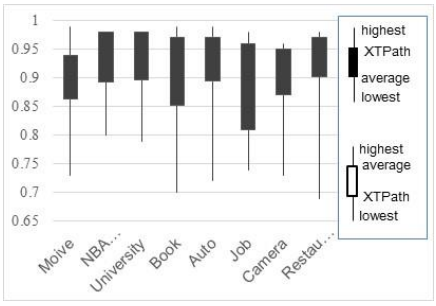

Fig. 3. The candle chart of XTPath system

3.4 Similarity Experimental Results on Unseen Dataset

A scenario for the proposed method is to find new patterns for new relations. That means when given a tag sequence pair belongs to an unknown relation, the similarity module should predict it as a genuine pair.

Take the vertical "*University*" for example, if we trained the similarity module on the relations set "*name*", "*phone*" and "*website*", it's easy to predict the genuine pair from same observed relations, say "*name*" or "*phone*". It's also easy to predict the impostor pair which consists a tag sequence from the observed relation and a tag sequence from an unknown relation. But it should be difficult to predict the genuine pair from the unknown relation, say a tag sequence pair from relation "*type*".

From the other hand, if the similarity calculation module robust enough, it can be used to find unknown relation. This can save much manual annotation work. For ex-

ample, we only annotate samples from a set of relations, say "*name*", "*phone*" and "*website*", and use the annotation data to train the similarity calculate module. If the similarity calculation module can find a cluster with high similarity inside and has low similarity with the observed relations. We can judge that these tag sequences belong to a new relation with high possibility.

To evaluate the performance, we conduct a set of experiments on unseen dataset. The unseen dataset means the dataset which contains the unknown relations. For each vertical, we made a n-fold cross validation (here n is the number of relations). Take the vertical "*University*" for example, we make 4-fold cross validation. We take one of the 4 relations ("*name*", "*phone*", "*website*" and "relation") as the test relation each time. Then we trained the similarity calculation module on the remained 3 relations and evaluated it on the selected test relation.

Table 5 reports the experimental results. Compared with the results on the observed relations, the performance on unseen relation is lower than it. However, the performance is not significant low. Among these verticals, the "*Restaurants*" got the highest performance with EER 0.019 (the EER on the observed data is: 0.01). It shown that the similarity calculation method is rather robust.

Table 5. Similarity Evaluation Results on Unseen Dataset

Vertical	**EER**	**Vertical**	**EER**
Autos	0.065	Movie	0.123
Book	0.160	NBA Player	0.109
Cameras	0.132	Restaurants	0.019
Jobs	0.163	University	0.089

4 Conclusion

This paper presents a new Web data extraction method with seed samples. The proposed method uses tag sequence of the XPath to build the extraction pattern. A Siamese network is adopted to learn the similarity metric from the tag sequence of seed samples. The proposed method builds the extraction patterns with the seed samples and continually refines it by finding more and more samples using the similarity metric. It can also be used to detect extraction pattern for the new relations.

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

References

1. Gurav, S., Gilani, J., Gore, V., et al.: Web Content Extraction Using Machine Learning. meta, 2018, 5(04)..
2. 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.

3. Ji, G., Liu, K., He, S. et al.: Distant Supervision for Relation Extraction with Sentence-Level Attention and Entity Descriptions, AAAI. 2017: 3060-3066.
4. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web, 2017, 8(3): 489-508.
5. W3C, XML Path Language (XPath) Version 1.0, W3C Recommendation, 1999, http://www.w3.org/TR/xpath.
6. Flesca, S., Furfaro, F. XPath query relaxation through rewriting rules. IEEE transactions on knowledge and data engineering, 23(10), 1583-1600.
7. Lockard, C., Dong, X. L., Einolghozati A, et al.: CERES: Distantly Supervised Relation Extraction from the Semi-Structured Web. arXiv:1804.04635, 2018.
8. Kushmerick, N., Weld, D. S., Doorenbos, R.: Wrapper induction for information extraction. In IJCAI, 1997.
9. 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.
10. 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.
11. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In KDD, 2014.
12. Gentile, A. L., Zhang, Z., Ciravegna, F.: Early steps towards web scale information extraction with lodie. AI Magazine, 36:55–64, 2015.
13. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. Advances in neural information processing systems. 2014: 2177-2185.
14. Mikolov, T.,Sutskever, I., Chen, K.: Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems. 2013: 3111-3119.
15. Bromley, J., Guyon, I., LeCun, Y.: Signature verification using a "siamese" time delay neural network. Advances in Neural Information Processing Systems. 1994: 737-744.
16. 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.
17. Cohen, J. P., Ding, W., Bagherjeiran, A.: Semi-supervised web wrapper repair via recursive tree matching. CoRR,abs/1505.01303, 2015.
18. 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.
19. Ortona, S., Orsi, G., Buoncristiano, M., Furche, T.: Wadar: Joint wrapper and data repair. PVLDB, 8:1996–1999, 2015.
20. Bronzi, M., Crescenzi, V., Merialdo, P., Papotti, P.: Extraction and integration of partially overlapping web sources. PVLDB, 6:805–816, 2013.