Incremental Knowledge Acquisition Approach for Information Extraction on both Semi-structured and Unstructured Text from the Open Domain Web

Maria Myung Hee Kim
Defence Science Technology Group
Edinburgh, SA 5111, Australia
maria.kim@dst.defence.gov.au

Abstract

Extracting information from semi-structured text has been studied only for limited domain sources due to its heterogeneous formats. This paper proposes a Ripple-Down Rules (RDR) based approach to extract relations from both semi-structured and unstructured text in open domain Web pages. We find that RDR's 'case-by-case' incremental knowledge acquisition approach provides practical flexibility for (1) handling heterogeneous formats of semi-structured text; (2) conducting knowledge engineering on any Web pages with minimum start-up cost and (3) allowing open-ended settings on relation schema. The efficacy of the approach has been demonstrated by extracting contact information from randomly collected open domain Web pages. The rGALA system achieved 0.87 F1 score on a testing dataset of 100 Web pages, after only 7 hours of knowledge engineering on a training set of 100 Web pages.

1 Introduction

Open Information Extraction (Open IE) (Banko et al., 2007; Wu and Weld, 2010; Fader et al., 2011) was introduced to extract information from the open domain Web where the relations of interest cannot be pre-defined in advance due to its heterogeneity in domain. Its purpose is to avoid specifying target relations and developing extraction models for individual target relations. The Open IE systems focus on discovering a binary relation candidate tuple in the form of (**E1**, **RelText**, **E2**) by identifying two entities of interest **E1** and **E2**, and the salient textual cues **RelText** (*aka 'relational text')* between the two entities. Then, they classify whether any binary relation **R** exists between the two entities in a given tuple to extract a binary relation tuple like (**E1, R, E2**).

To date extracting information from the open domain Web has mainly focused on the unstructured text (i.e., where text is formatted in paragraphs and expressed as full sentences). However, most Web pages generally contain information expressed in semi-structured text including tables, lists, isolated words or text snippets as well as unstructured text. Therefore, it is important to develop an IE capability that is able to process both semi-structured and unstructured text from the open domain Web.

Unlike unstructured text, semi-structured text usually includes HTML tags, which is primarily for formatting purposes. The variability in the way people use HTML tags impedes IE process. Above all, HTML tags have the following characteristics which hinder the IE task:
(1) HTML's tabular structure is often abused to arrange the graphical aspect instead of using cascading style sheets; and
(2) HTML tags can be deeply nested mixing relevant content with web noise in a loose manner.

Processing semi-structured text from the open domain Web is very challenging task as it needs to deal with heterogeneous formats as well as heterogeneous domains. It is difficult to create sufficient labelled data for semi-structured text in heterogeneous formats. Due to these difficulties, extracting information from semi-structured text has been studied only for specific domains (Chang et al., 2006). Moreover, existing semi-structured text IE approaches cannot be extended to the open domain Web sources as they usually require domain dependent inputs.

In summary, extracting information from semi-structured text in the open domain Web presents the following three main challenges:

Maria Myunghee Kim. 2017. Incremental Knowledge Acquisition Approach for Information Extraction on both Semi-structured and Unstructured Text from the Open Domain Web. In *Proceedings of Australasian Language Technology Association Workshop*, pages 88−96.

1. It is difficult to distinguish between the relevant content and web noise without domain knowledge. There is no explicit difference in HTML structure between them.
2. There are no clear linguistic markers (e.g., punctuation) to segment semi-structured text in the same manner as a *sentence* in unstructured text.
3. It is hard to create "sufficient" labelled training data and/or a complete ruleset for semi-structured text in open domain due to its heterogeneous formats.

Our rGALA system aims to extract information from both semi-structured and unstructured text in the open domain Web. To handle heterogeneous formats in semi-structured text, the rGALA system treats semi-structured text the same way as unstructured text in the Open IE task. The system filters out most of HTML tags and forms a binary relation candidate tuple (**E1**, **RelText**, **E2**); the system then extracts a binary relation tuple (**E1, R, E2**) if a relation R exists between the two given entities.

The rGALA system adopts a Ripple-Down Rules (RDR)' incremental knowledge acquisition approach; in RDR, the rule creation process is simple and rapid with ensured consistency in ruleset maintenance. The system does not require labelled training data or up-front knowledge for rule creation. Moreover, it supports open-ended settings on target relation definition by starting with a small set of relations and incrementally adding more relations as discovered during the extraction process.

2 Related Work

2.1 Open Information Extraction (Open IE)

Open Information Extraction (Open IE) aims to achieve domain-independent discovery of relations from the heterogeneous Web. Existing Open IE systems can be categorised into two groups based on the level of sophistication of the NLP techniques applied: (1) shallow syntactic parsing; and (2) dependency parsing. Shallow syntactic parsing based Open IE systems annotate sentences with Part-of-Speech (POS) tags and phrase chunk tags, then identify relations by matching patterns over these tags. The systems in this category include TextRunner (Banko et al., 2007), WOE*pos* (Wu and Weld, 2010), ReVerb (Fader et al., 2011) and R2A2 (Etzioni et al., 2011). Dependency parsing based Open IE systems utilise a dependency parser to identify whole subtrees connecting the relation predicate and its arguments. The systems in this category include OLLIE (Mausam et al., 2012), ClausIE (Corro and Gemulla, 2013), Wanderlust (Akbik and Brob, 2009), WOE*parse* (Wu and Weld, 2010) and KrakeN (Akbik and Loser, 2012). Each of these systems makes use of various heuristics to obtain extractions from the dependency parses. They are generally more time consuming than the shallow parsing based systems. They trade efficiency for improved precision and recall.

2.2 Ripple-Down Rules (RDR)

The basic idea of RDR (Compton and Jansen, 1990) is that each case is processed by the system and when the outcome is incorrect or NULL, one or more rules are added to provide the correct outcome for that case. The system also stores *cornerstone cases*, cases which triggered the creation of new rules.

The RDR approach has been applied to a range of NLP applications. Pham and colleagues developed KAFTIE using the RDR approach to extract positive attributions from scientific papers (Pham and Hoffmann, 2004) and to extract temporal relations (Pham and Hoffmann, 2006). KAFTIE was noted to have outperformed machine learning based systems. The RDR Case Explore (RDRCE) system (Xu and Hoffmann, 2010) combined RDR with a Machine Learning method. RDRCE was applied for POS tagging task and achieved a slight improvement over a state-of-the-art POS tagging system after 60 hours of knowledge engineering. A hybrid RDR-based Open IE system (Kim and Compton, 2012) makes use of RDR's incremental knowledge acquisition technique as an add-on to the state-of-the-art ReVerb Open IE system. With this wrapper approach, the ReVerb system's performance is further improved using RDR's error correction for the domain of interest.

2.3 IE systems for Semi-structured Text

Early IE systems for semi-structured text have been studied largely with manual approaches (Hammer et al., 1997; Arocena and Mendelzon, 1999) and supervised approaches (Kushmerick, 1997; Hsu and Dung, 1998; Soderland, 1999; Muslea et al., 1999; Califf and Mooney, 1999; Freitag, 2000; Laender et al., 2002). In order to increase the level of automation and reduce manual efforts, most of recent work has focused on semi-supervised approaches (Chang and Lui, 2001; Chang and Kuo, 2004) and unsupervised

approaches (Crescenzi et al., 2001; Arasu and Garcia-Molina, 2003; Zhai and Liu, 2005; Liu et al., 2010; Grigalis, 2013). Semi-supervised and unsupervised IE systems can be applied only to template based Web pages as they depend heavily on the existence of a common template (Chang et al., 2006).

In the same manner as the rGALA system, WHISK (Soderland, 1999) also aims to extract information from both semi-structured and unstructured text; but unlike rGALA, it targets specific domain Web pages and uses a supervised learning algorithm. To reduce the amount of manual labelling, WHISK interleaves learning new rules and annotating new instances (training examples) using selective sampling; thus, the learning and annotation process is iterative. It begins with an empty set of rules and at each iteration: (1) it presents to the user a batch of instances to be labelled via a graphical interface; (2) the labelled instances are added to a training set; (3) for each instance in a training set (not covered by the existing ruleset), WHISK learns the new rule using top-down induction, i.e., it finds the most general rule that covers the seed, then specialises the rule by adding terms incrementally until a stopping condition is met and finally (4) it prunes the rules.

3 rGALA System

3.1 rGALA Implementation

The rGALA system consists of the following three main components: (1) Preprocessor, (2) Tuple Extractor, and (3) RDR Engine.

(1) Preprocessor consists of the following four tools:

(a) Web transformer

A simple HTML transformation tool was built using JSOUP [1] to extract both semi-structured and unstructured text. To keep all potential information while minimising the amount of Web noise, the Web transformer tool conducts the following two steps:

Step1: removes most of HTML tags and attributes except <table>, <list> and <p> tags.

Step2: extracts text within <p> tags.

(b) Text segmenter

A text segmenter was built using the JFlex [2] (fast lexical analyser generator for Java) parser to identify text segments from both semi-structured and unstructured text. It takes a specification with a set of regular expressions and corresponding actions to identify a whole block of text for semi-structured text and a sentence for unstructured text.

(c) Tokeniser

Similar to the text segmenter a tokeniser was built using the JFlex parser to tokenise formal and informal multi-lingual text. It tokenises text based on its *semantic bearing* instead of white spaces. For example, a phone number in text such as *(08) 999 8888* is tokenised as a single token.

(d) generic Active Learning Application (gALA) system

The gALA [3] system identifies Part-of-Speech tags and Named Entity tags. The gALA incremental learning system is based on the Maximum Entropy (MaxEnt) algorithm. It is configurable and portable across domains with minimal or no NLP knowledge.

(2) Tuple Extractor extracts candidate tuples, [ENTITY_1, BETWEEN, ENTITY_2], which become *RDR cases* for binary relation classification task in the RDR Engine. A candidate tuple consists of two entities (ENTITY_1 and ENTITY_2) and a relational text (BETWEEN), which includes all the words between the two entities. The maximum number of tokens in the relational text is not limited by default but this value is configurable.

(3) RDR Engine follows these three steps:

(a) Step 1: The user checks the Relation Extraction (RE) result returned from the system.

For each RDR case, the system can return a correct or an incorrect RE result, or a NULL result when no rule was fired for the given case.

(b) Step 2: The user creates an RDR rule when the result returned is not correct or NULL.

If the system returns an incorrect RE result, a new rule is created under the rule which returned the incorrect result. If the system returns a NULL result, a new rule is created under the root rule.

(c) Step 3: The system evaluates the newly created RDR rule and the user refines it when required.

For the newly created rule, the system automatically evaluates it against the relevant cases (the parent rule's cases and the sibling

[1] https://jsoup.org/

[2] http://jflex.de/ - The JFlex parser uses Deterministic Finite Automata (DFA) to segment a text stream based on a set of user-defined rules.

[3] The gALA system was developed by Defence Science and Technology (DST) group.

rules' cases) in the system, which may conflict with the new rule. If the rule conflicts with these cases, the user can refine the rule's condition to make the rule more precise.

3.2 RDR Rule Description

An RDR rule has one or more conditions connected with an 'AND' operation, and a conclusion. Figure 1 shows the components of the RDR rule condition and conclusion. Note that '*Cond*' and '*Conc*' refer to 'Condition' and 'Conclusion' respectively.

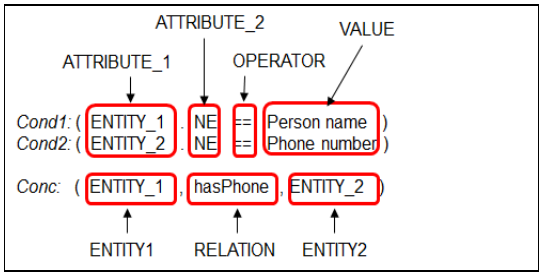

Figure 1: Components of RDR rule

(1) A *condition* consists of four components in the form of (**ATTRIBUTE_1.ATTRIBUTE_2 OPERATOR VALUE**).

(a) **ATTRIBUTE _1** refers to one of the 5 sections of a given text segment which is in the form of [E1BEFORE, ENTITY_1, BETWEEN, ENTITY_2, E2AFTER]. E1BEFORE and E2AFTER sections contain all the remaining tokens before the ENTITY_1 and after the ENTITY_2 sections, respectively.

(b) **ATTRIBUTE _2** refers to one of the NLP features; currently the following three NLP features are available:

- Lexical feature: token (TKN)
- Syntactic feature: Part-Of-Speech (POS)
- Semantic feature: Named Entity (NE)

(c) Currently the rGALA system supports nine **OPERATOR**s including '==', '*!=*', '*contains*', '*!contains*', '*regEx*', '*startsWith*', '*hasWordIn*', '*Pattern*' and '*NULL*'. Especially, the operator '*regEx*', '*hasWordIn*' and '*Pattern*' assist a single rule to handle multiple cases with similar patterns and words.

(d) **VALUE** is usually derived automatically in the system's GUI based on the choice made for the ATTRIBUTE_1 and ATTRIBUTE_2.

(2) A *conclusion* contains the relation extraction result in the form of (**ENTITY1**, **RELATION**, **ENTITY2**).

3.3 Rule Construction Example in Multiple Classification RDR (MCRDR)

A Multiple Classification RDR (MCRDR) is an 'n-ary' tree structure with only *except* edges. A case is evaluated by passing it to the root rule, which is always satisfied. An MCRDR evaluates all the first level rules which are direct children of the root rule. When a rule is satisfied, all its corresponding children rules are tested recursively where the children rules' conclusions overwrite the parent rule's conclusion. The inference process stops when there are no more children rules to evaluate.

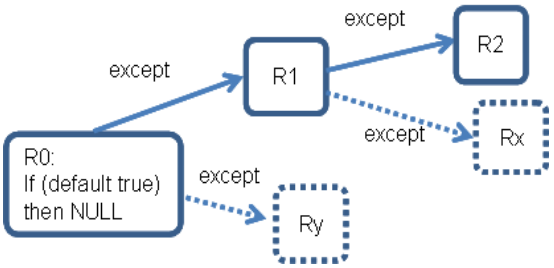

Figure 2: Rule construction example in MCRDR

The rGALA system applies an MCRDR for the single classification task as it is proved to be more efficient than a Single Classification RDR (SCRDR) even for single classification task (Kang et al., 1995). Figure 2 demonstrates MCRDR ruleset construction starting with an empty ruleset and the following three RDR cases described in the examples below.

Example 1. From the below text segment 1, RDR case 1 is identified. As the system returns a NULL result, a new rule is created.

Text segment 1: *'Copies can be bought by contacting: Dr. John Smith at NCID Edinburgh and phone: 77778888.'*
RDR case1: [**ENTITY_1: *'John Smith'***, BETWEEN: *'at NCID Edinburgh and phone:'*, **ENTITY_2: *77778888***]
RDR actions: 1. The default rule **R0** is fired with a NULL result as there is no rule to handle the given case. 2. A user creates a new rule **R1** under **R0** to extract 'hasPhone' relation from the given case. **R1**: *Cond1*:(ENTITY_1.NE == Person Name) AND *Cond2*:(ENTITY_2.NE == Phone Number) AND *Cond3*:(BETWEEN.NE !contains Person Name) *Conc*: (ENTITY_1, hasPhone, ENTITY_2)

Example 2. From the below text segment 2, RDR case 2 and RDR case 3 are identified. The system returns a correct result for the RDR case 2, but an incorrect result for the RDR case 3.

Text segment 2: '* Dr. Jane Smith * * Postal Address * * Elizabeth, East Ave., Australia * * Phone: (618) 322 4444, Fax: (618) 3115555 '*
RDR case2: [**ENTITY_1: '*Jane Smith*',** BETWEEN: '* Postal Address * * Elizabeth, East Ave., Australia * * Phone:*', **ENTITY_2: '*(618) 322 4444*'**]
RDR actions: 1. The rule **R1** is fired and returns the **[ENTITY_1, hasPhone, ENTITY_2]** result which is correct. The given case is saved under the rule **R1** and no further action is required.
RDR case3: [**ENTITY_1: '*Jane Smith*',** BETWEEN: '* Postal Address * * Elizabeth, East Ave., Australia * * Phone: (618) 322 4444, Fax:*', **ENTITY_2: '*(618) 311 5555*'**]
RDR actions: 1. The rule **R1** is fired and returns the [**ENTITY_1, hasPhone, ENTITY_2**] result which is an incorrect result as the ENTITY_2 is a fax number rather than a phone number. 2. The user needs to create an exception rule **R2** under **R1** to extract 'hasFax' relation from the given case by adding one more condition to specify the given case and returns the [**ENTITY_1, hasFax, ENTITY_2**] result. R1's three conditions become pre-conditions of R2 automatically. **R2**: *Cond1*: (BETWEEN.TKN contains '*Fax'*) *Conc*: (ENTITY_1, hasFax, ENTITY_2)

In RDR's exception rule structure, a user needs to select only a few conditions which are enough to distinguish the current case from the cornerstone case of the parent rule.

3.4 rGALA Graphic User Interface (GUI)

Figure 3 presents the RDR Engine GUI of the rGALA system. The GUI allows a user to view each RDR case and the system's classification results, and to form a rule when required. The numbers in figure 3 describes the followings:

1. Displays a text segment;
2. Displays identified candidate tuples in the form of [ENTITY_1, BETWEEN, ENTITY_2];
3. Displays relation extraction result returned from the RDR ruleset in the form of [ENTITY1, RELATION, ENTITY2];
4. Displays NLP features in the form of [E1BEFORE, ENTITY_1, BETWEEN, ENTITY_2, E2AFTER] for the current case, cornerstone case and evaluated cases;
5. Selects rule's conditions and a conclusion;
6. Displays rule's pre-conditions, conditions and a conclusion; and
7. Displays the process log, the currently fired rule's path and the evaluation results.

4 Experiments

The experiments were conducted to demonstrate the efficacy of the rGALA system in creating rules and the effectiveness of its ruleset on both semi-structured and unstructured text in open domain Web pages.

4.1 Experiment Settings

In order to examine the efficacy of the rGALA system on open domain Web pages, a set of Web pages was collected from various educational institutions (e.g. '*.edu*'), commercial companies (e.g. '*.com*') and government organisations (e.g. '*.gov*') web sites based on their URL addresses (without domain specific keywords). Manual annotation of a gold standard data is the very time consuming process. Therefore, from 1351 collected Web pages, only two sets of 100 Web pages were randomly selected as training and testing datasets without duplication.

Five types of relations about contact information including '*hasPhone*', '*hasFax*', '*hasAddress*', '*hasEmail*' and '*hasDomainAddress*' were chosen as initial target relations because: (1) they are commonly observed information in both semi-structured and unstructured text in open domain Web pages; and (2) they are usually written in heterogeneous formats influenced by personal, organisational and cultural preferences.

In the experiments, these five types of relations were further categorised into ten target relations. For example, the '*hasPhone'* relation was further specified into two relations '*O_hasPhone*' and '*P_hasPhone*' to capture the different entity types (organisation and person).

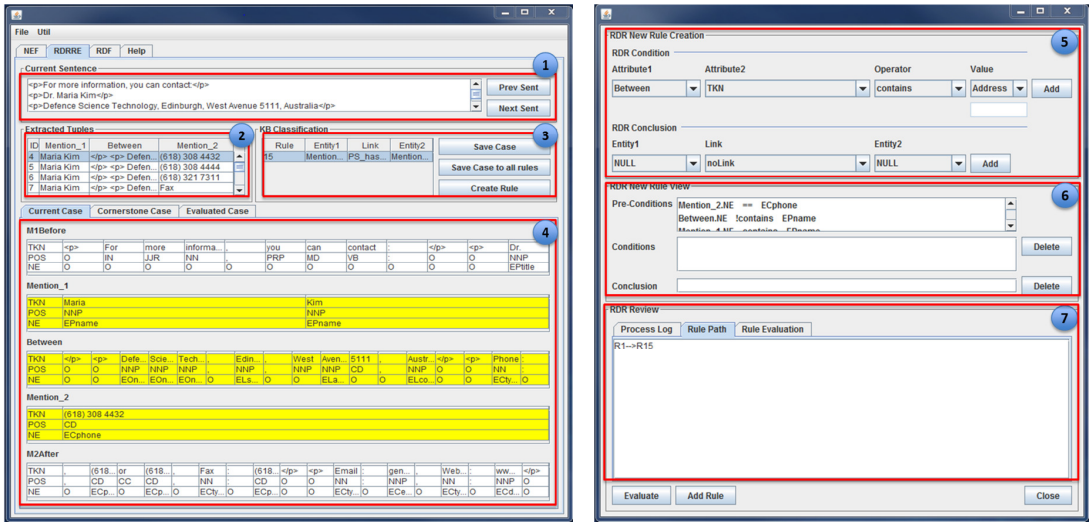

Figure 3: RDR Engine GUI for case-by-case incremental Knowledge Acquisition

Note that partially matched entities (esp. phone numbers) were evaluated as correct extractions in these experiments. For example, for '*+61 2 9999 5444/9999 1111*', the extracted entity '*9999 1111*' instead of '*+61 2 9999 1111*' was counted as a correct extraction.

4.2 Initial RDR Ruleset Construction

This section presents how the initial RDR ruleset was created to handle instances of the target relations from an empty ruleset.

To create the gold standard data, we manually analysed the post-processed 100 Web pages in the training dataset. As shown in the Table 1, a total of 325 instances of target relations were identified; 320 instances were found from semi-structured text written in 61 patterns and 5 instances were found from unstructured text written in 5 patterns.

To build the initial RDR ruleset, the rGALA system processed the 100 Web pages in the training dataset. It identified 1396 text segments and 5770 candidate tuples (*RDR cases*). In the tuple extraction process, 7 NE types were identified as entities' of interest including person, organisation, location, phone number, fax number, email address, postal address, and domain address.

The 5770 candidate tuples include 318 instances of the target relations and missed 7 target relation instances due to errors in detecting some phone number formatting. For example, for '*Phone: (618) 322 4444/5555*', the current system can detect '*(618) 322 4444*' but it cannot detect the last shortened phone number '*5555*'.

Table 1 shows the number of RDR rules created for each type of target relations. In total, 22 rules were created; 21 rules were created to cover 67 patterns and 1 rule was created to reshape an overly generalised rule causing a false positive error. On average, for semi-structured text, one rule covered three or more patterns.

Some RDR rules created for unstructured text also handled semi-structured text, and vice versa. These are indicated using bold numbers in Table 1. For example, no rule was required to handle the 17 instances of '*P_hasPhone*' relation from semi-structured text as it was covered by one rule created from one instance in unstructured text. This arises because the rGALA system handles semi-structured text the same way as unstructured text; it filters out most of HTML tags and identifies candidate tuples in the form of (**E1**, **RelText**, **E2**).

The knowledge engineering of 5770 candidate tuples (*RDR cases*) took about 7 hours without any extra-preparation time for labelling data or understanding the data structure in advance. The initial RDR ruleset construction time starts when a case is called and finishes when a rule is accepted as complete. This construction time is logged automatically.

	Semi-structured text			Unstructured text		
	Ins	**Pat**	**RDR**	**Ins**	**Pat**	**RDR**
O_hasPhone	149	13	4	**1**	**1**	**0**
P_hasPhone	**17**	**4**	**0**	1	1	1
O_hasFax	92	11	4	0	0	0
P_hasFax	4	3	1	0	0	0
O_hasAddress	22	12	3	0	0	0
P_hasAddress	10	4	1	0	0	0
O_hasEmail	18	9	2	1	1	1
P_hasEmail	6	3	1	0	0	0
O_hasDomain Address	**1**	**1**	**0**	2	2	2
P_hasDomain Address	1	1	1	0	0	0
Total	320	61	17	5	5	4

Table 1: rGALA rule creation analysis on the training Dataset. ('*Ins*' and '*Pat*' refers to '*Instances*' and '*Patterns*' respectively)

	Both Semi-structured and Unstructured text			
	Ins	**P**	**R**	**F1**
O_hasPhone	74	0.87	0.78	0.82
P_hasPhone	0	0	0	0
O_hasFax	49	1.00	0.86	0.92
P_hasFax	0	0	0	0
O_hasAddress	13	1.00	0.85	0.92
P_hasAddress	0	0	0	0
O_hasEmail	5	0.8	0.8	0.8
P_hasEmail	0	0	0	0
O_hasDomain Address	0	0	0	0
P_hasDomain Address	0	0	0	0
Total	141	0.93	0.83	0.88

Table 2: The rGALA performance on the testing dataset. ('*Ins*' refers to '*Instances*')

4.3 rGALA System Performance

This section presents the performance of the rGALA system on the 100 Web pages in the testing dataset with the RDR ruleset constructed from the training dataset.

To create the gold standard data, the post-processed 100 Web pages in the testing dataset were also manually analysed. As shown in Table 2, a total of 141 instances of the target relations were identified; 137 instances were found from semi-structured text written in 26 patterns and 4 instances were found from unstructured text written in 4 patterns. Among the 26 patterns from semi-structured text, 10 patterns were the same as the patterns from semi-structured text in the training dataset.

As shown in Table 2, the testing dataset only included four target relations out of our ten target relations including '*O_hasPhone'*, '*O_hasFax*', '*O_hasAddress*' and '*O_hasEmail*' due to the random selection of testing data. The testing dataset contained four out of five types of relations about contact information including '*hasPhone*', '*hasFax*', '*hasAddress*' and '*hasEmail*'.

When processing the testing dataset, the rGALA system identified 1386 text segments and 2818 candidate tuples (*RDR cases*). Overall, the rGALA system achieved reasonable and balanced performance of 0.88 F1 score with 0.93 precision and 0.83 recall. Total of 24 errors occurred including 4 False Positive (FP) errors and 20 False Negative errors (FN). All the 24 errors were caused from NE errors in the pre-processing phase; the 4 FP errors were due to incorrect NE types and the 20 FN errors were due to missed NEs. Among the 20 FN errors, the 8 FN errors were from missing shortened phone numbers format and 12 FN errors were from missing person and organisation named entities.

5 Discussion

As mentioned in section 4, the rGALA system achieved reasonable performance of 0.88 F1 score (with 0.93 precision and 0.83 recall) after only 7 hours of knowledge engineering on 100 open domain Web pages. No extra time was spent in analyzing the data, validating the rules or debugging.

In our experiment, the training dataset by chance contained more examples and patterns than the testing dataset. If the testing dataset were to contain more examples and patterns, the system may degrade. However, the rGALA system can quickly handle those uncovered examples in the testing dataset by adding rules incrementally.

The rGALA system cleans out HTML tags and treats semi-structured text in the same way as unstructured text. This approach brings out two main advantages shown in Table 1: (1) the rGALA system can handle various patterns of semi-structured text without any prior knowledge of the data structure/format and (2) its RDR rules work on both semi-structured and unstructured text. It is usually difficult to

perfectly extract information from open domain Web pages in one go. Subsequent maintenance and evolution of the ruleset is of utmost importance. In the rGALA system, a new rule is automatically organised in an exception structure, with automatic checking for any potential conflicts. This effectively addresses the critical maintenance issue from which most manual approaches suffer.

Although the size of the experimental dataset was not large, it fully satisfied our initial scenario where IE is required from a collection of open domain Web pages without prior knowledge of the data. Experience suggests that knowledge acquisition with RDR remains very simple and rapid even for large rulesets with over 10,000 rules (Compton et al., 2011).

As mentioned in section 2.3, WHISK (Soderland, 1999) also aimed for information extraction from both semi-structured and unstructured text. While the rGALA system builds one ruleset which works for both semi-structured and unstructured text for open domain sources, WHISK builds separate rulesets for semi-structured and unstructured text; it requires specific inputs for different domains such as the exact phrase delimiters to be extracted from semi-structured text.

The rGALA system is simple but effective; its case-by-case incremental knowledge acquisition approach helps to efficiently capture human knowledge to handle heterogeneous formats of semi-structured text in the open domain Web without prior knowledge, a labelled dataset or pre-defined relation schema. Rules can be updated as errors are uncovered, or when new formats are discovered, or new target relations are defined. The rGALA system is not a system to extract *all* potential relations from the *whole* Web, but it is a system to extract *any* relations of interests from *any* given Web pages. To date no work has been published on IE from semi-structured text for open domain Web pages. We have demonstrated that treating semi-structured text the same way as unstructured text for this problem shows considerable promise.

References

Alan Akbik and Jugen Brob. 2009. Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns. In *Proceedings of the 18th International conference on World Wide Web,* pages 6-15.

Alan Akbik and Alexander Loser. 2012. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52-56.

Arvind Arasu and Hector Garcia-Molina. 2003. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 337-348.

Gustavo O. Arocena and Alberto O. Mendelzon. 1999. WebOQL: Restructuring documents, databases and Webs. *Theory and Practice of Object Systems*, 5(3): 127-141.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670-2676.

Mary Elaine Califf and Raymond J. Mooney. 1999. Relational Learning of Pattern-Match Rules for Information Extraction. In *Proceedings of the 16th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence*. pages 328-334.

Chia-Hui Chang and Shih-Chien Kuo. 2004. OLERA: Semisupervised Web-data extraction with visual support. *IEEE Intelligent Systems*, 19(6): 56-64.

Chia-Hui Chang, M. Kayed, M.R. Girgis and K.F. Shaalan. 2006. A Survey of Web Information Extractio Systems. *IEEE Transactiona on Knowledge and Data Engineering*, 18(10): 1411-1428.

Chia-Hui Chang and Shao-Chen Lui. 2001. IEPAD: informatio extraction based on pattern discovery. In *Proceedings of the 10th International World Wide Web Conference*, pages 681-688.

Paul Compton and Bob Jansen. 1990. A philosophical basis for knowledge acquisition. *Knowledge Acquisition*, 2(3): 241-258.

Paul Compton, Lindsay Peters, Timothy Lavers, Yang-Sok Kim. 2011. Experience with long-term knowledge acquisition. In *Proceedings of the 6th international conference on Knowledge capture*, pages 49-56.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE: Clause-Based Open Information Extraction. In *Proceedings of the 22nd International conference on World Wide Web,* pages 355-366.

Valter Crescenzi, Giansalvatore Mecca and Paolo Merialdo. 2001. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *Proceedings of the 27th International Conference on Very Large Data Basaes*, pages 109-118.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the Conference on Artificial Intelligence*, pages 3-10.

Anthony Fader, Stephen Soderland and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535-1545.

Dayne Freitag. 2000. Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, 39(2-3): 169-202.

Tomas Grigalis, Towards web-scale structured web data extraction. 2013. In *Proceedings of the 6th ACM International Conference on Web search and data mining,* pages 753-758.

Joachim Hammer, Jason McHugh and Hector Garcia-Molina. 1997. Semistructured Data: The TSIMMIS Experience. In *Proceedings of the 1st East-European conference on Advances in Databases and Information Systems,* pages 22-22.

Chun-Nan Hsu and Ming-Tzung Dung. 1998. Generating finite-state transducers for semi-structured data extraction from the Web. *Information systems*, 23(8): 521-538.

Byeong Ho Kang, Paul Compton and Phil Preston. 1995. Multiple classification ripple down rules: evaluation and possibilities. In *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop*.

Myung Hee Kim and Paul Compton. Improving Open Information Extraction for Informal Web Documents with Ripple-Down Rules. 2012. In *Proceedings of the 12th Pacific Rim conference on Knowledge Management and Acquisition for Intelligent Systems*, pages 160-174.

Nicholas Kushmerick. 1997. Wrapper induction for information extraction. PhD dissertation, University of Washington.

Alberto H.F. Laender, Berthier Ribeiro-Neto and Altigran S. da Silva. 2002. DEByE - Data Extraction By Example. *Data and Knowledge Engineering,* 40(2): 121-154.

Wei Liu, Xiaofeng Meng and Weiyi Meng. 2010. Vide: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 22 (3), pages 447-460.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland and Oren Etzioni. 2012. Open Language Learning for Information Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Langauge Learning*, pages 523-534.

Ion Muslea, Steve Minton and Craig Knoblock. 1999. A hierarchical approach to wrapper induction. In *Proceedings of the 3rd Annual Conference on Autonomous Agents.* pages 190-197.

Son Bao Pham and Achim Hoffmann. 2004. Extracting Positive Attributions from Scientific papers. In *Proceedings of the 7th International conference on Discovery Science Conference*, pages 169-182.

Son Bao Pham and Achim Hoffmann. 2006. Efficient Knowledge Acquisition for Extracting Temporal Relations. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pages 521-525.

Stephen Soderland. 1999. Learning Information Extraction Rules for Semi-structured and Free Text. *Machine Learning*, pages 1-44.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118-127.

Han Xu and Achim Hoffmann. 2010. RDRCE: Combining Machine Learning and Knowledge Acquisition. In *Proceedings of the 11th International Workshop,* pages 165-179.

Yanhong Zhai and Bing Liu. 2005. Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on World Wide Web.* pages 76-85.