Improving Open Information Extraction for Informal Web Documents with Ripple-Down Rules

Myung Hee Kim and Paul Compton

The University of New South Wales, Sydney, NSW, Australia
{mkim978,compton}@cse.unsw.edu.au

Abstract. The World Wide Web contains a massive amount of information in unstructured natural language and obtaining valuable information from informally written Web documents is a major research challenge. One research focus is Open Information Extraction (OIE) aimed at developing relation-independent information extraction. Open Information Extraction systems seek to extract all potential relations from the text rather than extracting a few pre-defined relations. Existing Open Information Extraction systems have mainly focused on Web's heterogeneity rather than the Web's informality. The performance of the REVERB system, a state-of-the-art OIE system, drops dramatically as informality increases in Web documents.

This paper proposes a Hybrid Ripple-Down Rules based Open Information Extraction (Hybrid RDROIE) system, which uses RDR on top of a conventional OIE system. The Hybrid RDROIE system applies RDR's incremental learning technique as an add-on to the state-of-the-art REVERB OIE system to correct the performance degradation of REVERB due to the Web's informality in a domain of interest. With this wrapper approach, the baseline performance is that of the REVERB system with RDR correcting errors in a domain of interest. The Hybrid RDROIE system doubled REVERB's performance in a domain of interest after two hours training.

Keywords: Ripple-Down Rules, Open Information Extraction.

1 Introduction

The Web contains a large amount of information mainly in unstructured text and its quantity keeps increasing exponentially to an almost unlimited size. Web information extraction (WIE) systems analyze unstructured web documents and identify valuable information, such as particular named entities or semantic relations between entities. WIE systems enable effective retrieval of Web information to support various applications such as Automatic Text Summarization (ATS), Information Retrieval (IR) and Question-Answering (QA) systems.

The Web IE task has a number of significant differences compared to the traditional IE task of extracting particular instances from a small range of well-written documents. Most Web documents are not written under strict supervision and tend to be written informally. The followings are some characteristics of Web documents which affect extraction:

D. Richards and B.H. Kang (Eds.): PKAW 2012, LNAI 7457, pp. 160–174, 2012.

Informal Writing Styles. Huge amounts of Web documents are written informally and do not following strict writing styles like journalistic text [1]. Many NER techniques as part of a WIE rely on title and trigger words. As these markers are often absent in Web documents, there can be significant errors.

Spelling Mistakes and Incomplete Sentences. Web documents often include spelling mistakes and incomplete sentences, which hinder the syntactic analysis and cause extraction errors, since most of the existing systems are trained with formal texts with an assumption that the content of texts follows strict writing guidelines.

Large Amount of Newly and Informally Generated Vocabulary. Web documents contain a large number of newly generated unknown words, informal slang and short abbreviations which cannot be found in the formal dictionaries that are often utilized.

Web IE seeks to extract a large number of facts from heterogeneous Web documents while traditional IE has focused on extracting pre-defined relationships from smaller numbers of domain-specific documents. Open IE differs from previous IE in that its goal is to avoid using pre-defined target relations and extraction models for individual target relation. The OIE approach is intended to reduce the amount of time necessary to find the desired information. The open IE paradigm was proposed as 'preemptive IE' [2]. TextRunner [3] is an example of Open IE applied to Web IE.

Most OIE systems are developed using Machine Learning (ML) approaches and require a large amount of training data. They use self-supervised learning which generates a labeled training dataset automatically with some heuristics. For example, TextRunner uses an NLP tool to label entities and a parser to identify positive/negative examples with a small set of hand-written heuristic rules. A limit with this approach is that it cannot handle NLP errors since it relies on prior automatic labeling from NLP tools. This seriously affects the system performance as mentioned in [4], for example when a verb is incorrectly tagged as noun. Current OIE systems tend to use well-written journalistic documents as training data, probably to minimize errors from the NLP tools they depend on. It is likely that such training data is not the most appropriate for Web IE.

We have recently demonstrated how we can build an RDR-based OIE system that outperformed a previous machine-learning OIE system, TEXTRUNNER on Web data in a narrow range of interest [5]. Although the RDROIE system has not been tested on data outside the range of interest, necessarily it will perform worse a general OIE system in general domain. Therefore, we suggest that if we build an RDR-based OIE system to correct the errors of a more general system then overall it should produce better results because the minimum performance should be that of the general system performance.

Our contributions are summarized as follows:

- We propose the Hybrid RDROIE system that employs Ripple-Down Rules' incremental learning technique as an add-on to the state-of-the-art REVERB system in order to handle any performance degradation of REVERB due to the Web's informality.
- We evaluate the state-of-the-art REVERB system on a Web dataset with a fair level of Web informality and analysed errors that critically degrade performance.

- We demonstrate how the Hybrid RDROIE system handles informally written Web documents and doubles the performance of the REVERB system in a domain of interest after two hours training.

The remainder of this paper is structured as follows. Section 2 presents related work and section 3 presents an error analysis of the REVERB system on Web data. Section 4 explains our Hybrid RDROIE system in detail, section 5 presents the experimental setting and results and section 6 discusses the results and future work.

2 Related Work

2.1 Open Information Extraction

Sekine [6] introduced a new paradigm "On-Demand Information Extraction (ODIE)" which aims to eliminate high customization cost from target domain change. The ODIE system automatically discovers patterns and extracts information on new topics the user is interested in, using pattern discovery, paraphrase discovery, and extended named entity tagging. Shinyama et al. [7] developed the 'preemptive IE' framework with the idea of avoiding relation specificity. They clustered documents using pairwise vector-space clustering, and then they re-clustered documents based on named entity types in each document cluster. The system was tested on limited size corpora, because the two clustering steps made it difficult to scale the system for Web IE. TextRunner is the first open IE system for Web IE [3]. Two versions have been developed. The first is called O-NB which treated OIE task as a classification problem using a Naïve Bayes classifier [3]. The more recent system is O-CRF, which which treated the task as a sequential labeling problem using 'Conditional Random Fields (CRF)' [4]. O-CRF outperforms O-NB almost doubling recall. StatSnowball [8] performs both relation-specific IE and open IE with a bootstrapping technique which iteratively generates weighted extraction patterns. It employs shallow features only such as part-of-speech tags. In StatSnowball, two different pattern selection methods are introduced: l1-norm regularized pattern selection and heuristic-based pattern selection. Wu et al. [9] introduced a Wikipedia-based Open Extractor (WOE) which used heuristic matches between Wikipedia infobox attribute values and corresponding sentences in the document for self-supervised learning. WOE applied two types of lexical features: POS tag features and dependency parser features. Although with dependency parser features the system ran more slowly, it outperformed the system with POS tag features. Fader et al. [10] presented the problems of state-of-the-art OIE systems such as the TEXTRUNNER system [4] and the WOE system [9] where system outputs often contain uninformative and incoherent extractions. To address these problems, they proposed two simple syntactic and lexical constraints on binary relations expressed by verbs. Furthermore, the REVERB system proposed by Fader et al. is a 'relation first' rather than an 'arguments first' system, to try to avoid the errors of previous systems. REVERB achieved an AUC[1] that is 30% higher than WOE*parse* and more than double the AUC AUC of WOE*pos* or TEXTRUNNER [10].

[1] Area Under the Curve computed by a precision-recall curve by varying confidence threshold.

2.2 Ripple-Down Rules (RDR)

The basic idea of RDR is that cases are processed by the knowledge based system and when the output is not correct or missing one or more new rules are created to provide the correct output for that case. The knowledge engineering task in adding rules is simply selecting conditions for the rule which is automatically located in the knowledge base with new rules placed under the default rule node for newly seen cases, and exception rules located under the fired rules. The system also stores cornerstone cases, cases that triggered the creation of new rules. If a new rule is fired by any cornerstone cases, the cornerstones are presented to the expert to select further differentiating features for the rule or to accept that the new conclusions should apply to the cornerstone. Experience suggests this whole process takes at most a few minutes. A recent study of a large number of RDR knowledge bases used for interpreting diagnostic data in chemical pathology, showed from logs that the median time to add a rule was less than 2 minutes across 57,626 rules [11].

The RDR approach has also been applied to a range of NLP applications. For example, Pham et al. developed KAFTIE, an incremental knowledge acquisition framework to extract positive attributions from scientific papers [15] and temporal relations that outperformed machine learning [16]. Relevant to the work here, RDR Case Explorer (RDRCE) [17] combined Machine Learning and manual Knowledge Acquisition. It generated an initial RDR tree using transformation-based learning, but then allowed for corrections to be made. They applied RDRCE to POS tagging and achieved a slight improvement over state-of-the-art POS tagging after 60 hours of KA. The idea of using an RDR system as a wrapper around a more general system was suggested by work on detecting duplicate invoices where the RDR system was used to clean up false positive duplicates from the general system [12].

3 Error Analysis of REVERB on a Web Dataset

In this section, we analyse the performance of the REVERB system on a Web dataset, Sent500 and categorise the types of errors. The experiment is conducted on the Web dataset referred to as 'Sent500' and the detail of it is explained in section 5.1. Originally, in this dataset, each sentence has one pair of entities manually identified for the relation extraction task, but tags are removed for this evaluation. That is, there are no pre-defined tags in the Sent500 dataset used here.

Extractions are judged by the following: Entities should be proper nouns; pronouns such as he/she/it etc. are not treated as appropriate entities. In a tuple, entity1, relation and entity2 should be located in the appropriate section. For example, if entity1 and relation are both in entity1 section and the relation section is filled by noise then, it is treated as an incorrect extraction. On the other hand, if entity1, relation and entity2 are properly located, then some extra tokens or noise are allowed as long as they do not affect the meaning of extraction. For example, the tuple extraction (**Another example of a statutory merger** , **is** , **software maker Adobe Systems acquisition of Macromedia**) is incorrect but the tuple extraction (**Adobe** , **has announced the acquisition of** , **Macromedia)** is correct. N-ary relations such as (**Google** , **has officially acquired YouTube for** , **$ 1.65 bil**) are treated as a correct extraction.

Table 1. The performance of the REVERB system on the Sent500

	Total	VERB	NOUN+PREP	VERB+PREP	INFINITIVE
P	41.32%	69.72%	42.03%	69.86%	50.00%
R	45.25%	55.62%	26.13%	54.26%	20.45%
F1	43.20%	61.88%	32.22%	61.08%	29.03%

Table 1 shows the performance of the REVERB system overall and on four different classes. The overall result is evaluated based on all extractions from Sent500 using REVERB, while the four category results are evaluated based on extraction of the pre-tagged entities and relations in Sent500. The results show that overall REVEB performance on Sent500 is quite poor at around 40%. The VERB and VERB+PREP categories show higher precision than the NOUN+PREP and INFINITIVE categories. Especially, the recall of NOUN+PREP and INFINITIVE categories is very low, 26.13% and 20.45%, respectively. This is because that the REVERB system aims to extraction binary relations expressed by verbs.

Table 2. Incorrect extraction errors analysis on each category

	VERB	NOUN+PREP	VERB+PREP	INFINITIVE
Correct relation but incorrect entities	84%	18%	91%	33%
Correct relation and entities but incorrect position with noise	4%	27%	0%	0%
Incorrect relation and entities	12%	55%	9%	67%

Table 2 summaries the types of incorrect extraction errors on four categories. For VERB and VERB+PREP categories, most of false positive errors, 84% and 91% respectively, are due to incorrect entity detection while relation detection is correct. As REVERB extracts entities using noun phrases, which are located nearest to the detected relation, it often recognizes an inappropriate noun phrase as an entity.

For example, in a sentence '*Google has acquired the video sharing website YouTube for $ 1.65billion (883million) in shares after a large amount of speculation over whether __ was talking about a deal with __ .*', (**Google**, **has acquired**, **the Video**) is extracted instead of (**Google**, **has acquired**, **YouTube**).

Some of entities have boundary detection errors due to noise or symbols used within an entity. For instance, REVERB only extracted 'Lee' for an entity 'Tim Berners – Lee'. On the other hand, in the NOUN+PREP and INFINITIVE categories, most of false positive errors, 55% and 67% respectively, are due to incorrect detection of both relations and entities.

Table 3. Missed extraction errors analysis on each category

	VERB	NOUN+PREP	VERB+PREP	INFINITIVE
NLP error	72%	7%	14%	0%
Non-verb-based relation	11%	93%	5%	100%
Noise	11%	0%	0%	0%
Unusual expression	6%	0%	81%	0%

Table 3 presents the types of missed extraction errors on the four categories. In the VERB category, 72% of errors are caused by NLP errors. For example, in '*Google Buys YouTube.*', REVERB misses an extraction because 'Buys' is tagged as a noun.

Due to the Web's informality such as informally used capital letters, NLP tools often incorrectly annotate Web datasets. In the VERB+PREP category, 81% of the errors are due to unusual expressions. REVERB includes approximately 1.7 million distinct normalized relation phrases, which are derived from 500 million Web sentences. As REVERB uses this set of relation phrases to detect relations, it tends to miss relations not expressed in the system. For example, in the sentence '*Kafka born in Prague*', the relation 'born in' is not detected while in the sentence '*Kafka was born in Prague*', the relation 'was born in' is correctly detected. Moreover, in the sentence '*Google acquire YouTube*', the relation 'acquire' is not detected while in the sentence '*Google acquires YouTube*', the relation 'acquires' is correctly detected.

In the NOUN+PREP and INFINITIVE categories errors are mostly due to non-verb-based relation extraction. As REVERB aims to extract binary relations expressed by verbs, it only can extract NOUN+PREP and INFINITIVE type relations when there is verb before a NOUN+PREP and INFINITIVE relation phrase. That is, when there exist tuples like (entity1, verb NOUN+PREP, entity2) and (entity1, verb TO VB, entity2), REVERB can extract NOUN+PREP and INFINITIVE type relations in the Sent500 dataset. For example, a tuple (**Novartis** , **completes acquisition of 98% of** , **Eon Labs**) is successfully extracted from the sentence '*Novartis completes acquisition of 98 % of Eon Labs , substantially strengthening the leading position of its Sandoz generics unit (Basel , July 21 , 2005)*' while no tuple is extracted from the sentence '*Here is the video of the two _ founders talking about the Google acquisition in their YouTube Way !*' because there is no verb between two entities 'Google' and 'YouTube'. In the INFINITIVE case, for instance, a tuple (**Paramount Pictures** , **agreed to buy** , **DreamWorks SKG**) is correctly extracted from the sentence '*- Viacom s Paramount Pictures agreed to buy DreamWorks SKG for $ 1.6 billion in cash and debt , wresting the movie studio away from NBC Universal and securing the talents of Steven Spielberg .*', while no tuple is extracted from '*Adobe About to Buy Macromedia .*' because there is no verb between 'Adobe' and 'Macromedia'.

The REVERB system has shown very poor recall on Sent500. 89% and 95% of the false negative errors (which affects recall) on VERB and VERB+PERP are due to the Web's informality (NLP error, noise and unusual expression). Also, 93% and 100% of false negative errors on NOUN+PERP and INFINITIVE are due to non-verb relations. The aim of Hybrid RDROIE is to correct REVERB's.

4 Hybrid RDROIE System Architecture

The Hybrid RDR-based Open Information Extraction (Hybrid RDROIE) system shown in Fig. 1 consists of four main components: preprocessor, NLPRDR KB learner, REVERB system and TupleRDR KB learner. We considered that it was more efficient to clean up NLP errors before using REVERB rather than just fixing errors after. This is because as shown above, REVERB's recall is very poor and one of main reasons is NLP error. If we use REVERB first before NLPRDR KB, then we cannot improve REVERB's recall. In section 4.1, the implementation details of the three components are explained; the RDR rule syntax is described in section 4.2 and RDR KB construction demonstrated in section 4.3 and finally the user interface is shown in section 4.4.

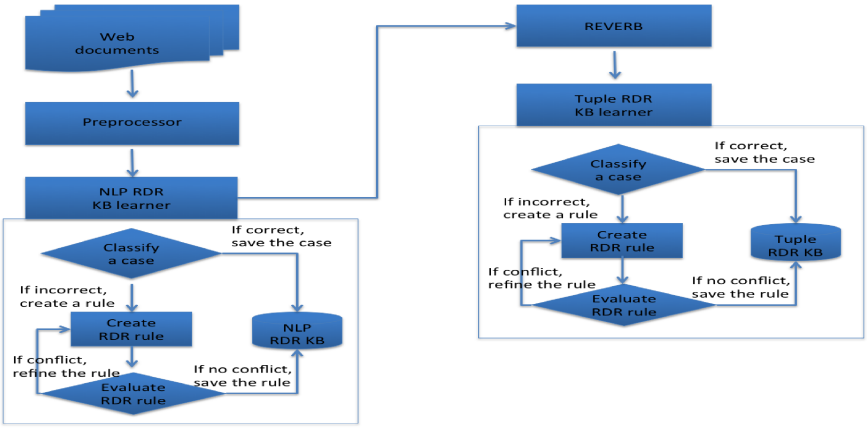

Fig. 1. Architecture of the Hybrid RDROIE system

4.1 Implementation

Preprocessor. The preprocessor converts raw Web documents into a sequence of sentences, and annotates each token for part of speech (POS) and noun and verb phrase chunk using the OpenNLP system. It also annotates named entity (NE) tags using the Stanford NER system. Annotated NLP features are used when creating rules.

NLPRDR KB Learner. The NLPRDR KB is built incrementally while the Hybrid RDROIE system is in use. The systems takes a preprocessed sentence as a case and the NLPRDR KB returns the NLP classification result. When the NLP classification result is not correct, the user adds exception rules to correct it. There are three steps:

Step1: NLP Classification. The NLPRDR KB takes each preprocessed sentence from the preprocessor and returns the classification results. If RDR rules are fired and the fired rules deliver correct the classification results, then the system saves the case (a sentence) under the fired rules. The system also saves the refined sentence based on

the fired rule's conclusion action and sets the current case sentence as the refined sentence and passes it to the REVERB system for tuple extraction. If the root rule is fired and the sentence is correct, then the current case sentence is kept as is.

Step2: Create RDR Rule. Whenever the NLPRDR KB gives incorrect classification results, the user adds rules to correct the classification results.

Step3: Evaluate and Refine RDR Rule. Once the new rule is created, the system automatically checks whether the new rule affects KB consistency by evaluating all the previously stored cornerstone cases that may fire the new rule. To assist the expert, the user interface displays not only the rule conditions of previously stored cases but also the features differentiating the current case and any previously stored cases, which also satisfy the new rule condition but have a different conclusion. The expert must select at least one differentiating feature, unless they decide that the new conclusion should apply to the previous case.

As the NLPRDR KB corrects NLP errors on the sentence, more tuples can be extracted from the REVERB system.

TupleRDR KB Learner. The TupleRDR KB is used to correct errors on REVERB's tuple extractions, whereas the NLPRDR KB described above was used to tidy up NLP errors on the given sentence before using REVERB.

The TupleRDR KB is built incrementally while the system is in use. In the Hybrid RDROIE system, the user gets the tuple extractions in the form of binary relation (entity1, relation, entity2) from the REVERB system. The TupleRDR KB returns the tuple classification result and if the tuple classification result is incorrect, the user adds exception rules to correct it. There are following three steps:

Step1: Tuple Classification. The TupleRDR KB takes each tuple extraction from the REVERB system and returns the classification results. If the RDR rules fire and the fired rules deliver the correct classification results, then the system saves the case (a tuple extraction) under the fired rules and also saves the corrected tuple based on the fired rules' conclusion action. If the root rule is fired and the tuple is correct, then only action is to save the correct extraction in the database.

Step2: Create RDR Rule. Whenever incorrect classifications results are given (by the REVERB system or the TupleRDR KB add-on), the user adds rules to correct the classification results.

Step3: Evaluate and Refine RDR Rule. Same as step3 in NLPRDR KB.

4.2 Hybrid RDROIE's Rule Description

An RDR rule has a condition part and a conclusion part: 'IF (*condition*) THEN (*conclusion*)' where *condition* may indicate more than one condition. A *condition* consists of three components: (ATTRIBUTE, OPERATOR, VALUE). In the NLPRDR KB, the ATTRIBUTE refers to the given sentence and in the TupleRDR KB, ATTRIBUTE refers to the given sentence and each element of the given tuple, ENTITY1, RELATION and ENTITY2. Both the NLPRDR KB and the TupleRDR KB provide 9 types of OPERATOR as follows:

- hasToken: whether a certain token matches
- hasPOS: whether a certain part of speech matches
- hasChunk: whether a certain chunk matches
- hasNE: whether a certain named entity matches
- hasGap: skip a certain number of tokens or spaces to match the pattern
- notHasPOS: whether a certain part of speech does not match
- notHasNE: whether a certain named entity does not match
- beforeWD(+a): checks tokens located before the given attribute token by +a
- afterWD(+a): checks tokens located after the given attribute token by +a

VALUE is derived automatically from the given sentence corresponding to the ATTRIBUTE and OPERATOR chosen by the user in the user interface.

In both NLPRDR KB and TupleRDR KB, conditions are connected with an 'AND' operator. A sequence of conditions begins with the 'SEQ' keyword and it is used to identify a group of words in sequence order, so patterns can be detected. For instance, the sequence condition: 'SEQ((RELATION hasToken '**born**') & (RELATION hasToken '**in**'))' detects 'born in' in the RELATION element of the tuple.

In NLPRDR KB, a rule's CONCLUSION part has the following form:

(fixTarget, --- target element
fixType, --- refinement type, default is token
fixFrom, --- classification result before refinement
fixTo) --- classification result after refinement

In TupleRDR KB, a rule's CONCLUSION part has the following form:

(relDetection, --- relation existence detection
fixTarget, --- target element
fixFrom, --- classification result before refinement
fixTo) --- classification result after refinement

4.3 Examples of Hybrid RDROIE Rules

The Hybrid RDROIE system is based on Multiple Classification RDR (MCRDR) [13]. Fig. 2 demonstrates MCRDR-based KB construction as the NLPRDR KB system processes the following three cases starting with an empty KB (with a default rule R1 which is always true and returns the NULL classification).

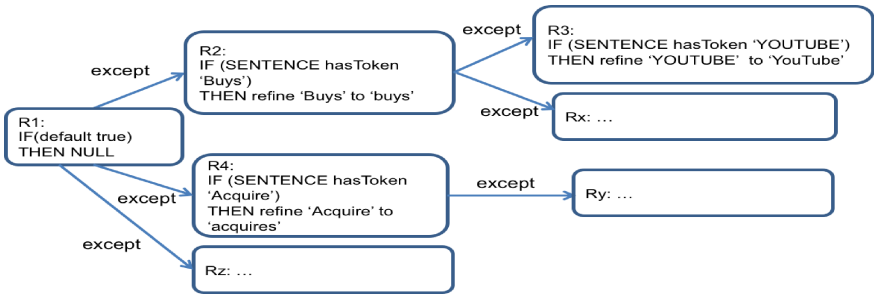

Fig. 2. MCRDR structure of the NLPRDR KB system

Case1: A sentence '*Google Buys YouTube.*'

→ The default rule R1 is fired and the KB system returns a NULL classification and the user considers this is an incorrect classification result because 'Buys' should be refined as 'buys'.
→ A user adds a new rule R2 under the default rule R1.
Case2: A sentence '*Google Buys YOUTUBE.*'
→ Rule R2 is fired but the user considers the result is incorrect because 'YOUTUBE' should be refined to 'YouTube' to be correctly tagged by NLP tools and extracted by REVERB.
→ A user adds an exception rule R3 under the parent rule R2.
Case3: A sentence '*Adobe system Acquire Macromedia.*'
→ Default rule R1 fires and the KB system returns a NULL classification, which the user considers as an incorrect result because 'Acquire' should be refined to 'acquires' to be extracted correctly by the REVERB system.
→ A user adds a new rule R4 under the default rule R1.

Fig. 3 shows an MCRDR based KB construction as the TupleRDR KB system processes the following three cases (tuples) starting with en empty KB.

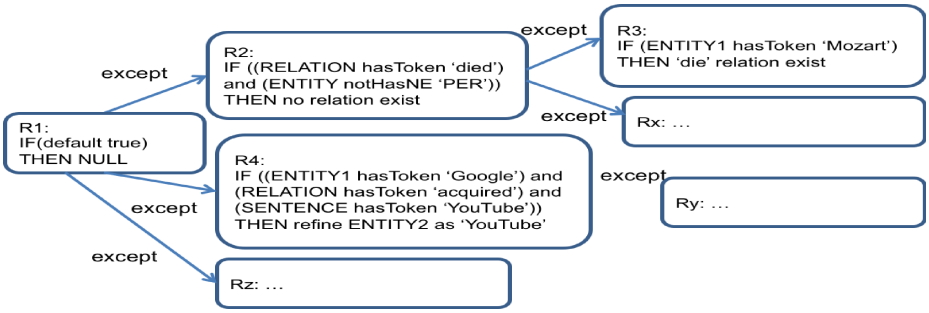

Fig. 3. MCRDR Structure of the TupleRDR KB System

Case1: Tuple (**Prague July 3, 1883** , **died near** , **Vienna June 3, 1924**) from the given sentence '*Franz Kafka (born Prague July , 1883 died near Vienna June 3, 1924) was a famous Czech - born , German - speaking writer .*'

→ The default rule R1 is fired and the KB system returns a NULL classification, which the user considers as an incorrect classification result because ENTITY1 contains 'Prague July 3, 1883' instead of 'Franz Kafka'.
→ A user adds a new rule R2 under the default rule R1.
Case2: Tuple (**Wolfgang Amadeus Mozart** , **died** , **5 December 1791**) from the given sentence '*Wolfgang Amadeus Mozart died 5 December 1791.*'
→ Rule R2 fires and classifies the given tuple as 'no relation tuple' and the user considers it as an incorrect result because the tuple contains a correct relation. This happens since the NE tagger has not tagged the token 'Mozart' as PERSON NE.
→ The user adds an exception rule R3 under the parent rule R2.

Case3: Tuple (**Google** , **has acquired** , **the Video**) from the given sentence '*Google has acquired the Video sharing website YouTube for $ 1.65billion (883million).*'
→ The default rule R1 fires and the KB system returns a NULL classification, which the user considers as an incorrect result because ENTITY2 contains 'the Video' instead of 'YouTube'.
→ A user adds new rule R4 under the default rule R1.

4.4 Hybrid RDROIE User Interface

The Hybrid RDROIE system provides a graphic interface that aids in creating and adding RDR rules and maintaining the KB system by end-users. Because most of the relevant values are displayed automatically and the system is built based on the normal human process of identifying distinguishing features when justifying a different conclusion, a user should be able to manage the system after few hours training. Industrial experience in complex domains supports this [11]. Fig. 4 presents the Hybrid RDROIE user interface. The Hybrid RDROIE system is written in Java (Java 1.6) and adopted the OpenNLP system (version 1.5), the Stanford NER system (version 1.5) and the REVERB OIE system (version 1.1).

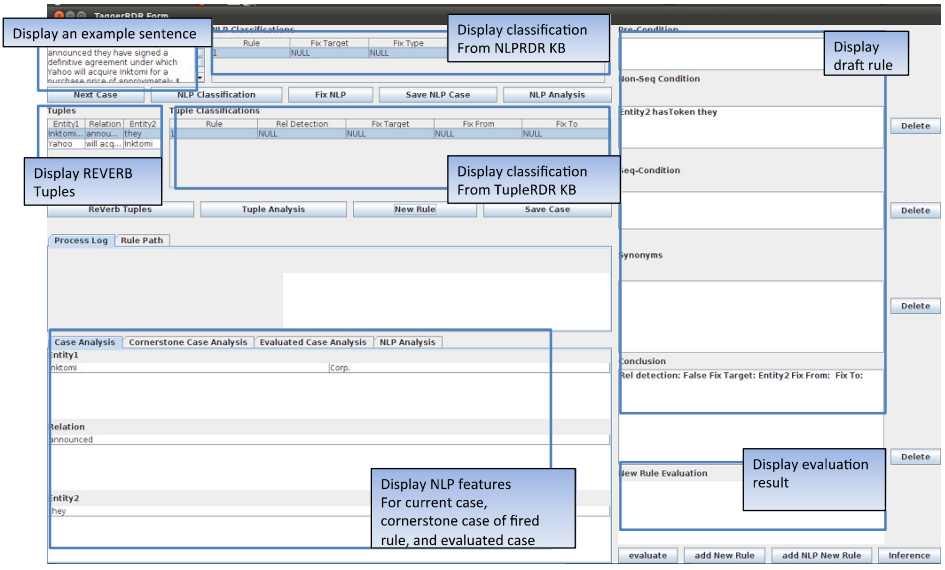

Fig. 4. User Interface of the Hybrid RDROIE system

5 Experiments

Section 5.1 describes the Web dataset used. Section 5.2 shows the initial knowledge base construction of the Hybrid RDROIE system and the section 5.3 presents the

results achieved by the Hybrid RDROIE system and discusses how our system improved the existing performance of the REVERB system on the Web data

5.1 Web Datasets

The experiments were conducted on the two Web datasets, Sent500 and Sent300, which were also used in experiments for the RDROIE system [5]. Sent300 was used as training dataset to construct the RDR KB and Sent500 was used as test dataset to test the performance of the overall Hybrid RDROIE system. Sent300 is derived from the MIL dataset developed by Bunescu and Mooney [14]. The MIL dataset contains a bag of sentences from the Google search engine by submitting a query string 'a1 ******* a2' containing seven wildcard symbols between the given pair of arguments. Sent500 was developed by Banko and Etzioni [4]. It contains some randomly selected sentences from the MIL dataset and some more sentences for 'inventors of product' and 'award winners' relations using the same technique as used for MIL datasets. In Sent300 and Sent500, each sentence has one pair of entities manually tagged for the relation extraction task, but those entity tags were removed in this experiment. That is, there are no pre-defined tags in our training and test dataset.

5.2 RDR Initial KB Constructions

This section presents the analysis of the initial KB construction using the Hybrid RDROIE system. In processing the Sent300 training set, 119 NLP errors were identified and rules were added as each error occurred. For the NLPRDR KB, 28 new rules were added under the default rule R1 and 6 exception rules were added for the cases which received incorrect classification results from earlier rules. Secondly, 98 tuples extracted from the REVERB system, which could not be corrected by fixing NLP errors with the NLPRDR KB were identified as incorrect relation extractions and rules were added for each incorrect tuple extraction. For the TupleRDR KB, in total, 14 new rules were added under the default rule R1 and 5 exception rules were added for the cases which received incorrect classification results from earlier rules.

As the Hybrid RDROIE system handles for both NLP error and tuple error, all rules are used together within a single process flow. In total 53 rules were added within two hours. KB construction time covers from when a case is called up until a rule is accepted as complete. This time is logged automatically.

5.3 Hybrid RDROIE Performance

The Hybrid RDROIE system was tested on the Sent500 dataset. Table 4 presents the performance of the Hybrid RDROIE system on total extractions and on four category extractions. The REVERB system extracts multiple tuples from a sentence without using pre-defined entity tags. The performance on total extractions is evaluated on all tuple extractions of the REVERB system. The performance on four extraction types is calculated based on the explicit tuples when the pre-defined entity tags exist.

Table 4. The performance of the Hybrid RDROIE system on total extraction and on four categories of extraction on the Sent500 dataset

	Total	VERB	NOUN+PREP	VERB+PREP	INFINITIVE
P	90.00%	90.24%	74.00%	90.00%	85.00%
R	81.45%	83.15%	66.67%	86.17%	77.27%
F1	85.51%	86.55%	70.14%	88.04%	80.95%

On total extractions, overall the Hybrid RDROIE system achieved 90% precision, 81.45% recall and an F1 score of 85.51%, while the REVERB system by itself achieved 41.32% precision, 45.25% recall and a 43.20% F1 score on the same dataset (see table 1). That is, the Hybrid RDROIE system improved the performance of the REVERB system by almost double. Precision improved as the TupleRDR KB reduced false positive errors by filtering incorrect extractions and recall improved as the NLPRDR KB reduced false negative errors by amending informal sentences.

Across the four category extractions, on average the Hybrid RDROIE system improved around 30% on precision, recall and F1 score over all four categories. VERB and VERB+PREP categories achieved high precision and NOUN+PREP and INFINITIVE categories also achieved reasonably good precision. In particular the recall of NOUN+PREP and INFINITIVE categories improved dramatically from 26.13% and 20.45% to 66.67% and 77.27%, respectively. This improvement suggests that the Hybrid RDROIE system supports relation extractions on non-verb expression while the REVERB system mainly extracts relation expressed by verbs.

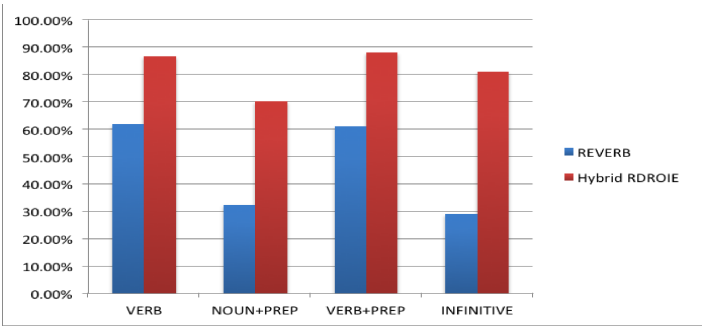

Fig. 5. Performance improvement of the Hybrid RDROIE system from the REVERB system on F1 score over four categories

Fig. 5 presents the performance improvement of the Hybrid RDROIE system over the REVERB system on F1 score over four categories. For all categories the Hybrid RDROIE system improved REVERB performance. In particular, NOUN+PREP and INFINITIVE category had the biggest improvement.

6 Discussion

Table 4 shows that the Hybrid RDROIE system achieves high precision and recall after only two hours initial KB construction by a user on a small training dataset. Given the very rapid training time we suggest that rather than simply having to accept an insufficient level of performance delivered by the REVERB system in a particular application area, it is a worthwhile and practical alternative to very rapidly add rules to specifically cover the REVERB system's performance drop in that application area.

In the Hybrid RDROIE system, lexical features are mainly utilized when creating RDR rules. Because it is difficult to handle the Web's informality using NLP features such as part-of-speech, chunk phrase and named entity most of errors for the REVERB system occurred because of NLP errors on the Web dataset.

The advantage of utilising lexical features directly was demonstrated by the REVERB system's performance compared to previous OIE systems such as the TEXTRUNNER and WOE systems [4, 9], but we note that this was for data without a high level of informality. The REVERB system primarily utilises direct lexical feature matching techniques using the relation phrase dictionary, collected from 500 million Web sentences. Previous OIE systems such as the TEXTRUNNER system and the WOE system utilised more NLP features such as part-of-speech and chunk phrase and used machine learning techniques on a large volume of heuristically labelled training data (e.g. 200,000 sentences used for TEXTRUNNER and 300,000 sentences for WOE). The Hybrid RDROIE system, similarly utilises lexical features to handle the Web's informality and improve the REVERB system's performance further. In consequence, as shown in table 4, the Hybrid RDROIE system outperformed the REVERB system and achieved a good balanced overall result compared to other OIE systems. Section 4.3 showed examples of the Hybrid RDROIE system using lexical features in rule creation. We note that other systems focusing more on NLP issues outperformed REVERB on this data set, but we also note that as shown in [5] a pure RDR approach did even better.

The Hybrid RDROIE system is designed to be trained on a specific domain of interest. One might also comment that the rules added are simple fixes of lexical errors, and to produce a large system would need a large number of rules. This is really the same type of approach as REVERB with its vast relation phase dictionary. If the Hybrid RDROIE system is to be used for a particular domain, which we believe would be the normal real world application, we see little problem in adding the rules required and keeping on doing this as new errors are identified and we note that in pathology people have developed systems with over 10,000 rules [11]. The Hybrid RDROIE system required very little effort and the study here it took about two minutes on average to build a rule. Experience suggests that knowledge acquisition with RDR remains very rapid even for large knowledge bases [11]. On the other hand, if the aim was a very broad system, it would also be interesting to see if it was possible to extend domain coverage by some type of crowd sourcing, with large numbers of people on the web contributing rules.

References

1. Collot, M., Belmore, N.: Electronic Language: A New Variety of English. In: Computer-Mediated Communications: Linguistic, Social and Cross-Cultural Perspectives (1996)
2. Shinyama, Y., Sekine, S.: Preemptive information extraction using unrestricted relation discovery. In: Proceedings of the HLT/NAACL (2006)
3. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence (2007)
4. Banko, M., Etzioni, O.: The Tradeoffs Between Open and Traditional Relation Extraction. Paper Presented at the Proceedings of ACL 2008: HLT (2008)
5. Kim, M.H., Compton, P., Kim, Y.-s.: RDR-based Open IE for the Web Document. In: 6th International Conference on Knowledge Capture, Banff, Alberta, Canada (2011)
6. Sekine, S.: On-demand information extraction. In: Proceedings of the COLING/ACL (2006)
7. Shinyama, Y., Sekine, S.: Preemptive information extraction using unrestricted relation discovery. In: Proceedings of the HLT/NAACL (2006)
8. Zhu, J., Nie, Z., Liu, X., Zhang, B., Wen, J.-R.: StatSnowball: a statistical approach to extracting entity relationships. In: Proceedings of the 18th WWW (2009)
9. Wu, F., Weld, D.S.: Open Information Extraction using Wikipedia. In: The 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden (2010)
10. Fader, A., Soderland, S., Etzioni, O.: Identifying Relations for Open Information Extraction. In: EMNLP, Scotland, UK (2011)
11. Compton, P., Peters, L., Lavers, T., Kim, Y.-S.: Experience with long-term knowledge acquisition. In: 6th International Conference on Knowledge Capture, pp. 49–56. ACM, Banff (2011)
12. Ho, V.H., Compton, P., Benatallah, B., Vayssiere, J., Menzel, L., Vogler, H.: An incremental knowledge acquisition method for improving duplicate invoices detection. In: Proceedings of the International Conference on Data Engineering (2009)
13. Kang, B., Compton, P., Preston, P.: Multiple classification ripple down rules: evaluation and possibilities. In: Proceedings of the 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, February 26-March 3, vol. 1, pp. 17.1 – 17.20 (1995)
14. Bunescu, R.C., Mooney, R.J.: Learning to Extract Relations from the Web using Minimal Supervision. In: Proceedings of the 45th ACL (2007)
15. Pham, S.B., Hoffmann, A.: Extracting Positive Attributions from Scientific Papers. In: Discovery Science Conference (2004)
16. Pham, S.B., Hoffmann, A.: Efficient Knowledge Acquisition for Extracting Temporal Relations. In: 17th European Conference on Artificial Intelligence, Italy (2006)
17. Xu, H., Hoffmann, A.: RDRCE: Combining Machine Learning and Knowledge Acquisition. In: Kang, B.-H., Richards, D. (eds.) PKAW 2010. LNCS, vol. 6232, pp. 165–179. Springer, Heidelberg (2010)