Ripple-Down Rules based Open Information Extraction for the Web Documents

Myung Hee Kim

School of Computer Science and Engineering

The University of New South Wales

Australia

Supervised by

Emeritus Professor Paul Compton

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy (Computer Science and Engineering)

April 2012

ORIGINALITY STATEMENT

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed …….

Date ……..

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 few pre-defined relations.

Previous machine learning-based Open Information Extraction systems require large volumes of labelled training examples and have trouble handling NLP tool errors caused by the Web's informality. These systems used self-supervised learning methods that generate a labelled training dataset automatically using NLP tools with some heuristic rules. As the number of NLP tool errors increase because of the Web's informality, the self-supervised learning-based labelling technique produces noisy labels and critical extraction errors.

The thesis first presents Ripple-Down Rules based Open Information Extraction (RDROIE) an approach to open information extraction that uses Ripple-Down Rules (RDR)' incremental learning technique. The RDROIE system is trained on a particular domain and demonstrates it outperforms TEXTRUNNER and StatSnowball on other test data from the same domain after 2.5 hours training.

A Hybrid RDROIE system was introduced which applies RDR method 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 reduce the performance degradation of the REVERB system due to the Web's informality in a domain of interest. With

this wrapper approach, the baseline performance would be that of the conventional system with RDR correcting errors from the baseline system in a domain of interest. The Hybrid RDROIE system demonstrates it doubles REVERB's performance in a domain of interest after 2 hours training.

Finally the wrapper approach was applied to Named Entity Recognition (NER) task and a Ripple-Down Rules based Named Entity Recognition (RDRNER) system is constructed that uses RDR's incremental learning technique as an add-on to the state-of-the-art Stanford NER system to address performance degradation problems due to the Web's informality. The RDRNER system demonstrates it improves the Stanford NER's degraded performance on informal Web text in a domain of interest after 4 hours training.

In summary this thesis demonstrates that an incremental learning approach with humans providing rules can be used to deal with errors that are difficult for more automated approaches. It also demonstrates that the incremental approach, used on top of a more automated system, can improve the performance of the more general automated systems. This leads to the general suggestion that if an incremental learning approach is used on top of the best current system, the performance of the overall system should be better than the state of the art general systems.

Acknowledgements

I would like to say Thank you very much to all who surrounded and supported me during my PhD work.

From the bottom of my heart, I would like to express my special gratitude to my supervisor Emeritus Prof. Paul Compton for the great deal of his support and mentorship. His guidance with marvelous knowledge always inspired and motivated me. Sincerely I shall be grateful to Prof. Compton all my life for his effort, time, trust, and patience.

My work was also supported a great deal by the academic staffs in the School of Computer Science and Engineering. My acknowledgements also go to the staffs at the school.

Above all, I would like to express my dearest gratitude to my parents, brother and my husband Hiroshi for their understanding, love, patience, encouragement and support, for which I will not be able to fully acknowledge.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Web and Open Information Extraction

The Web contains a vast amount of valuable information mostly in unstructured natural language and the amount of information keeps growing exponentially. Web information extraction (WIE) systems analyse unstructured Web documents and extract important information, such as particular named entities or semantic relations between entities. WIE systems facilitate 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 Information Extraction (WIE) task has a number of significant differences compared to the traditional information extraction task of extracting particular instances from small range of formally written documents. Web IE seeks to extract a large number of facts from

heterogeneous and informal Web documents while traditional IE focuses on extracting pre-defined relationships from smaller numbers of domain-specific documents.

Most Web documents are written under no strict supervision and likely to be written casually. There are a number of Web document characteristics that make the WIE task more challenging. Firstly, many Web documents are written freely not following strict writing styles like journalistic documents. Because strict writing markers which help to identify entities are often missing in informal Web documents, NLP tools which rely on such indicators cannot work efficiently and may even cause a significant numbers of errors. Secondly, Web documents often include spelling mistakes and incomplete sentences, which hinder the syntactic analysis of NLP tools and cause extraction errors. Lastly, Web documents may contain large amounts of newly generated unknown vocabularies and abbreviations, which are not included in formal dictionaries.

The Open Information Extraction (OIE) task differs from the traditional IE task in that its purpose is to avoid specifying target relations and extraction models for individual target relations. OIE is intended to reduce the amount of time necessary to find desired information from the Web with its ever increasing volume of valuable information. The OIE task focuses on discovering both the entities E1 and E2, believed to participate in a relation R, and the salient textual cues that provide evidence of the relation between two entities and extracting a binary relation tuple like (E1, R, E2). Most OIE systems try to extract binary relation tuples representing all possible relationships between entities with a single scan of the given documents. Thus, OIE systems run in time O(D), where D is the number of documents in the corpus while traditional IE systems must run in O(D*R), where R is the number of pre-defined relations as the extraction processes must repeat on the entire corpus every time the system is given a new set of target relations.

Previous Machine Learning (ML) based OIE systems like TEXTRUNNER (Banko and Etzioni, 2008) and WOE (Wu and Weld, 2010) require a large number of labelled training examples (e.g. 200,000 heuristically labelled for TEXTRUNNER and 300,000 for WOE). They use a self-supervised learning technique that generates a labelled training dataset automatically using a dependency parser to identify positive/negative examples with a small set of hand-written heuristic rules. The heuristic labelling of training examples avoids manual labelling effort but it produces noisy labels that results in overall OIE extraction errors. One of the critical limits with this approach is that it cannot handle NLP tool errors since it relies on prior automatic labelling from NLP tools. This seriously affects performance as mentioned in (Banko and Etzioni, 2008), for example when a verb is incorrectly annotated as a noun by the NLP tools due to the Web's informality, resulting in incorrect labels for training examples.

Previous ML-based OIE systems tended to use formal journalistic documents as training data, probably to minimize errors from the NLP tools they depended on. It is likely that such formally written training data is not the most appropriate dataset for a Web Information Extraction (WIE) task due to the Web's informal characteristics noted above.

1.2 Ripple-Down Rules (RDR)

The basic idea of RDR is that cases are processed by the Knowledge Based (KB) 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 benefit of the RDR technique is that the KB is built incrementally while the system is already in use. Experience suggests the whole process of manually adding new rule

takes less than 2 minutes per rule (Compton et al., 2011). Easy maintenance and validation are the most important reasons that RDR achieved commercial success.

Single Classification RDR (SCRDR) was successfully used in the medical expert system Pathology Expert Interpretative Reporting System (PEIRS) which was used for interpreting chemical pathology results (Compton et al 1992; Compton et al 1993). While SCRDR merely allows a single refinement to a rule, Multiple Classification RDR (MCRDR) allows multiple refinements supporting multiple conclusions for a case. Numbers of studies have shown that MCRDR can build more a compact and less redundant knowledge base faster than SCRDR even for single classification domains (Kang et al., 1995; Kang, 1996; Kang et al., 1998). MCRDR also requires fewer cases to be seen for covering multiple subdomains than SCRDR (Richards, 2009). In MCRDR it is possible to utilise various attribute parts of the case to generate different rules and conclusions for each area of focus or subdomain (Richards, 2009). Therefore, MCRDR the methodology has been chosen for our RDR-based OIE system.

The key contrast between the type of machine learning used in OIE and RDR is that machine learning requires many examples, whereas an expert can develop an RDR for a single example, contrasted with the examples the system already knows about. This means that for an OIE system, it might be possible to fix errors as they are identified. In particular, errors due to the Web's informality might be able to be corrected as they occur, rather than trying to accumulate sufficient examples for machine learning to work successfully.

It will be necessary to write rules based on text, so it should be noted that RDR have already been successfully applied to various Natural Langue Processing (NLP) areas such as information

extraction (Pham and Hoffmann, 2004a, 2004b, 2005 and 2006) and part-of-speech tagging (Xu and Hoffmann, 2010, Nguyen et al., 2011).

The central hypothesis of the thesis is that it is possible to monitor the performance of an OIE system and to write RDR-type rules to extract information more appropriately when required, and the resulting system will perform at a high level. There is a necessary limitation with such an approach that the system can be only expected to deal with the types of errors, that have already been seen and for which it has been trained. However, this is a different limitation from IE systems which can only deal with pre-specified entities and relations; with RDR the limitation is that the system can only deal with the types of OIE errors it has experienced. The thesis proposes how this approach can be used in conjunction with more conventional OIE systems to produce a more powerful outcome.

1.3 Thesis Contributions

1. The thesis first presents an RDROIE system, an approach to open information extraction that uses Ripple-Down Rules' incremental learning technique is constructed. The aim of this was to investigate the utility of RDR on OIE task. For this purpose,
    a. Experimental results on developing and evaluating an RDROIE system for a test domain are presented

b. The various factors of informal Web documents which cause errors in open information extraction from the Web are discussed and we demonstrate from the case study how the RDROIE approach handles these factors
c. We analyse the structure of the RDROIE knowledge base to investigate how different aspects of the system contribute to its performance

The RDROIE system will not necessarily work on new types of errors that it has not encountered in the training data, and so might not work well on different domains. Our second approach was to use RDR on top of a conventional OIE system, so that the baseline performance would be the conventional system with the RDR correcting errors from the baseline system in a domain of interest.

2. Hybrid RDROIE employs Ripple-Down Rules' incremental learning technique as an add-on to the state-of-the-art REVERB OIE system in order to extend domain coverage and handle any performance degradation of REVERB due to the Web's informality. To resolve this problem,
    a. The performance of the REVERB system on informal Web documents is analysed and error types that critically degrade performance are categorised
    b. We demonstrate how the Hybrid RDROIE system handles informally written Web documents and improves the performance of the REVERB system to almost double.

The performance of entity extraction in the RDROIE system was critically limited by the Stanford NER (Named Entity Recognition) system as its performance degraded significantly on informal Web documents. Therefore, the Stanford NER system needed to be improved to handle

Web's informality. As we have already introduced the idea of RDR on top of general system in the Hybrid RDROIE system, we investigated this approach for Web-scale named entity recognition task in the specific domain of interest.

3. RDRNER employs Ripple-Down Rules' incremental learning technique as an add-on to the state-of-the-art Stanford NER system in order to handle the problems of the Web's informality. To address this problem,
    a. The performance of the Stanford NER system on informal Web documents is analysed and error sources that critically degrade the performance are categorised.
    b. We demonstrate how the RDRNER system handles informally written Web documents and improves the performance of the Stanford NER system to be equivalent to its best performance on formal documents.

Finally we consider some possible areas of further development using an RDR approach for OIE.

1.4 Thesis Organisation

The rest of thesis is organised as follows. Chapter 2 provides a review of Information Extraction (IE) and its process and discusses Web Information Extraction (WIE) and Open Information Extraction (OIE) in depth. Chapter 3 reviews the literature on Ripple-Down Rules. Chapter 4 presents the RDROIE system, the initial RDR-based OIE system. Chapter 5 describes the Hybrid RDROIE system where RDR is used on top of a conventional REVERB OIE system to improve its performance in the domain of interest, but where the conventional system still provides a

baseline performance for errors for which it has not yet been trained. Chapter 6 presents the RDRNER system, where the same idea of a hybrid system is used to improve the NER system that RDROIE depend on. Chapter 7 concludes with possible directions for RDR-based systems for OIE and NER tasks.

Chapter 2. Information Extraction

Information Extraction (IE) should automatically identify relevant information from unstructured text and store it into a structured (machine-readable) format. Extracting instances of semantic relationships can be very useful for applications such as biography extraction, ontology construction and question answering.

The Web contains a vast amount of information mainly in an unstructured text form and its quantity keeps increasing exponentially to an almost unlimited size. Web information extraction (WIE) systems analyse unstructured Web documents and identify valuable information, such as particular named entities or semantic relations between entities.

Open Information Extraction (OIE) aims at developing relation-independent information extraction systems. OIE systems seek to extract all potential relations rather than extracting pre-defined target relations from the given documents.

Until recently, most studies in IE have focused on extracting pre-defined target relationships from small, domain-specific text corpora, such as newswire articles or job postings while WIE studies aim at extracting information from massive and heterogeneous Web text. Although recent WIE systems have demonstrated the ability to extract a large number of facts, these systems require pre-defined target relations and relevant training data for each target relation as input. Consequently, these systems' extraction process has to be repeated whenever a relation of interest is to be investigated. OIE greatly increases the speed of WIE and the scale of the output as it aims to extract all potential instances of relations without pre-defined target relations.

2.1 Basic Approaches

2.1.1 Knowledge Engineering

Rule-based methods were the main approach behind the first information extraction systems and are still the most widely used, in the area of information extraction and other natural language processing tasks (Appelt et al., 1993; Cunningham et al., 2001). Such approaches attempt to employ syntactic and semantic patterns to discover relations by using the knowledge of experts. The extraction rules should be sufficient to deal with linguistic diversity in a certain domain. This approach has major problems on poor adaptability and poor robustness, especially when dealing with large-scale or new domain data. For example, every time rules have to be re-written for different tasks or domains and cannot be reused. For each new domain the whole effort must be repeated, resulting in a high labour cost and time consumption, a problem known as the knowledge engineering bottleneck.

2.1.2 Machine Learning

In order to avoid the knowledge engineering bottleneck problem, machine learning (ML)-based methods that are data-driven have become increasingly popular. This approach enables systems to automatically improve performance on a task using statistical or symbolic learning methods and reduces labour cost and the time involved in developing systems. The ML approach is especially useful for domains where one does not know exactly how to solve a problem, but training datasets and experience are available. Information extraction is a particularly appropriate application domain for machine learning. IE systems need to handle very complex input data and the diversity of human language, which makes it difficult to predict all possible expressions.

Supervised learning techniques require large corpora as input, which has been annotated, usually manually, and output a learned model that can later be applied to new inputs. There are two main problems with this approach; one is that it costs time and labour to prepare sufficiently large and accurately annotated training data and the other is that the domain coverage of the annotated datasets will have its limits.

Since a large amount of annotated training data is usually hard to obtain, researchers turn their interest to unsupervised learning methods that learn models without the use of manually annotated inputs. Particularly because the Web is large and highly heterogeneous, unsupervised and self-supervised learning techniques became popular for Web Information Extraction (WIE). Bootstrapping-based methods have been developed for unsupervised learning for semantic relation extraction. Generally, such methods work by iteratively classifying unlabelled instances and adding confidently classified unlabelled instances into the labelled data using a model learnt from augmented labelled data in the previous loop. Brin (1998) proposed bootstrapping from a small initial set of seeds to utilise the duality between patterns and relations by collecting

common patterns from book titles and authors. In the SNOWBALL system, Agichtein et al. (2000) improved Brin's method by employing a pattern matching-based classifier. The TEXTRUNNER system (Banko et al., 2007) used self-supervised learning, which automatically labelled a set of candidate extractions applying a set of heuristic constraints. It relied on Web's redundancy to improve overall extraction accuracy.

2.2 Text Analysis

Information extraction systems can utilise useful features from different levels of text analysis. Various levels of analysis provide features from simple lexical (e.g. word-based) features to more complex features derived from grammatical or semantic analysis. Part-Of-Speech (POS) and phrase chunk features are provided by an intermediate levels of analysis.

2.2.1 Part-Of-Speech (POS)

A POS tagger takes a sequence of tokens as input, and labels each token with its appropriate tag such as verb, noun, adjective, determiner etc. As tokens may have multiple possible POS tags, the role of the POS tagger is to choose the most appropriate one. Most of the widely used POS taggers have been developed statistically using either supervised or unsupervised training. A supervised approach uses annotated corpora where each word is tagged with its correct POS. With an unsupervised approach, the corpora contain no such annotations; words are automatically clustered into groups that are then used for the learning process. Various POS taggers are available such as the Brill POS tagger (Brill, 1995), the Hepple POS tagger (Hepple, 2000) in the Gate framework (Cunningham et al., 2002). Since these systems are generally

trained on news corpora, they may need a little tuning when used in a different domain for the specific words only used in that domain. The state-of-the-art POS taggers have been shown to achieve accuracy rate of up to 97% precision and are generally considered to be robust (Toutanova et al., 2003). We note that Ripple-Down Rule-based taggers have also achieved the state of the art accuracy (Xu and Hoffmann, 2010; Nguyen et al., 2011).

2.2.2 Named Entity Recognition (NER)

Named Entity Recognition (NER) is one of the key tasks in various NLP applications such as Information Extraction (IE), Automatic Text Summarization (ATS), Information Retrieval (IR) and Question-Answering (QA) (Califf and Mooney, 1997; Rozenfeld and Feldman, 2008). It automatically identifies proper names in text and classifies them into a set of categories such as persons, geographical locations, names of organisations, dates, times, amounts of money and percentages. NER emerged as a specific information extraction task at the sixth Message Understanding Conferences (MUC-6) (Grishman and Sundheim, 1996). The MUCs have offered the opportunity to evaluate NER systems on the same data in a competition and provided a guideline for entity annotation.

NER has mainly adopted two approaches. One is referred to as a knowledge-based approach and uses explicit resources such as rules and gazetteers (dictionaries), which usually are handcrafted by experienced language experts (Rau, 1991; Ravin and Wacholder, 1996). This achieves good performance but the development can be very time-consuming. The other approach is learning-based and uses statistics or machine learning. Supervised learning techniques learn automatically on large amounts of annotated text corpora. (Bikel et al., 1999; Borthwick, 1999; Asahara and Matsumoto, 2003). While this approach does not need language engineering expertise, it requires large amounts of annotated training data. Such training corpora

are available from evaluation forums but there are limitations in the amount of annotated data and coverage of the domain. Recent studies have explored semi-supervised (Nadeau and Sekine, 2007) and unsupervised learning techniques (Alfonseca and Manandhar, 2002; Etzioni et al., 2005), which do not require annotated corpora.

Current NER systems are mainly trained on journalistic documents such as news articles. Consequently they have not been trained to deal with the informal Web documents, resulting in dramatic performance drops on informally written Web documents. For these reasons, some studies claim that NER is a major source of errors for Web Information Extraction (WIE) (Nguyen et al., 2007; Zhu et al., 2009). Recent WIE systems (Banko and Etzioni, 2008; Zhu et al., 2009) have avoided the NER process and instead utilised only shallow features such as part-of-speech (POS) tags and phrase chunk tags for entity extraction and relation extraction pattern generation and then relied on the Web's redundancy to improve accuracy of the systems. This approach has critical limitations on less redundant informal Web documents such as Wikipedia, blogs and news comment.

There have been two main approaches for Web-scale NER systems.

Relying on Large Web Resources. The main limitation of supervised learning (SL) approaches is the prerequisite of a large amount of annotated training data. The lack of such annotated corpora for Web documents and the excessive cost of creating annotated corpora to cover highly heterogeneous Web documents led to an alternative approach for scaling: semi-supervised learning (SSL) which relies on very large collections of Web documents and information redundancy (Zacharias, 2008). Currently, bootstrapping-based methods are the main research focus for semi-supervised learning for information extraction. Generally, they work by

iteratively classifying un-annotated instances and adding confidently classified instances into the annotated data using a model learnt from the augmented annotated data in the previous loop. Etzioni et al. (2005) utilised search engines to compute statistical properties and built the list of named entities from the whole Web as a corpus by bootstrapping a rule template with class key words and a small set of generic extraction patterns.

Constructing Gazetteers from Web Resources. As existing NER systems' gazetteers encounter vocabulary limitations when applied to highly heterogeneous Web documents, some studies have focused on automatic extraction of known entities from the Web or Wikipedia. Toral and Mu˜noz (2006) and Kazama and Torisawa (2007) constructed high coverage gazetteers from Wikipedia. Ratinov and Roth (2009) extracted 14 gazetteers from the Web which have high precision but low recall, so to improve coverage they extracted 16 gazetteers from Wikipedia which contains over 1.5M entities. This method costs much less in time and labour as gazetteers can be generated using automated or semi-automated techniques (Etzioni et al. 2005; Wang and Cohen, 2007). Nadeau et al. (2006) used such lists in an unsupervised NER system, and demonstrated outstanding performance on some of the MUC named entity tasks. Kazama and Torisawa (2007) also improved the performance of their supervised NER system by utilising a Wikipedia-based gazetteer. Although utilising gazetteers through lookup-based methods can be a useful technique in named entity recognition, it may cause a significant number of errors and relatively low recall due to its ambiguity and incomplete coverage, especially with heterogeneous Web documents. Mikheev et al. (1999) have argued that adding more gazetteers does not necessarily guarantee better performance in named entity recognition. Marrero et al (Marrero et al., 2009) evaluated modern NER systems and demonstrated that those NER systems which are highly reliant on gazetteers (Afner (Zaanen and Moll´a, 2007) and YooName (Nadeau

et al., 2006) systems) tend to show poor performance compared to other NER systems. They claimed that the reason is lack of context analysis in these systems.

2.2.3 Parser

One of the significant tasks in natural language processing is the development of parsers. The goal of a parser is to capture the structure of a single sentence. For a given input sentence, a parser returns a tree or directed graph that captures various levels of linguistic information, including the POS of each token, the presence of phrases, grammatical structures and semantic roles. Additional information such as the subject and object of a verb phrase may also be provided depending on the parsers. The linguistic information provided by parsers can be useful for identifying relationships between entities within a given sentence.

Some parsing approaches adopted declarative unification-based formalisms like lexical functional grammars (LFG) (Kaplan and Bresnan, 1982) or head-driven phrase structure grammars (HPSG) (Pollard and Sag, 1994). On the other hand, there are shallower approaches using a cascade of finite state transducers (Hobbs et al., 1997). These shallow methods only perform partial parsing for syntactic analysis. Ambiguities, such as prepositional phrase attachment, are not considered by shallow parsers resulting in them being more efficient and robust than complete and deep parsers.

One of the key challenges for a parser is domain-adaptability, the capability to analyse input from any domain. This is a difficult task as some inputs are not in the form of the parser's formal grammar, or fittingly represented in the parser's training data. A well-known broad-coverage English parser, MINIPAR (Lin, 1998) had demonstrated recall of 81.81% over a set of news corpora, but its recall dropped when applied to more complex genres such as fiction (75.29%),

memoirs/letters (77.15%) and scholarly prose (79.95%). Ratnaparkhi (1999) evaluated the cross-domain portability of a maximum-entropy parser and showed that when trained on a corpus of Wall Street Journal (WSJ) news and tested on sections of the Brown corpus, which contains a set of fiction, magazines and journal articles, precision decreased by 6.8% and recall by 6.2% compared to training and testing entirely on the WSJ. Gildea (2001) also presented similar results when studying a statistical chart parser using the same corpora, citing cross-domain reduction of precision by 5.8% and recall by 5.6%.

2.2.4 Lexico-Syntactic Patterns

One of the most widely used methods for automatic information extraction is lexico-syntactic pattern analysis. This was first proposed by Hearst (1992) to acquire hyponymy relations from large text corpora, and then followed by many successful systems, such as KnowItAll (Etzioni et al., 2005) and PANKOW (Cimiano et al., 2004; Cimiano et al., 2005).

Hearst (1992) assumed that the patterns almost always signify the relations of interest and the information extracted will not be erroneous. Some levels of abstraction employing lexico-syntactic patterns were used for the automatic acquisition of the hyponymy lexical relation. The following steps were applied to discover new patterns:

1. Decide on a lexical relation

2. Collect a list of terms for which this relation is known to hold

3. Find places in the corpus where these expressions occur syntactically near one another and note the environment

4. Analyse the similarities among these environments and generate common patterns

5. Choose new patterns and use them to collect more instances of the target relation and go back to Step2

Many researchers (Brin, 1998; Etzioni et al., 2005; Snow et al., 2005) were inspired by Heart's (1992)'s approach and attempted to automate step 3 and 4.

Bowden et al. (1996) have developed the domain independent system, KEP to extract intra-sentential conceptual relations such as definitions and exemplifications by using manually created patterns at different levels of abstraction. During the first two stages, only non-syntactic patterns over words were employed. Then, in the last stage, patterns over POS were utilised to capture syntactic constituents such as nouns or verbs. The system detects potential sentences with target conceptual relations using lists of positive and negative phrases through the first stage. For instance, sentences detected by positive phrases will then be excluded if they include negative phrases. In the second stage, the extraction process was conducted employing combinatorial pattern matching. Sentences nominated during the first stage are used to match against a list of patterns with named lexicons. For example, a pattern 'eX = X' where 'e' indicates a named lexicon which contains 'example', 'examples', 'an example', and 'an example of', '=' indicates a named lexicon which includes 'is' and 'X' matches any tokens. This pattern matches the following sentence: "An example of a high-level language is PASCAL." The first X ('a high-level language') will detect the concept and the second X ('PASCAL') will capture an example of the concept. In the third state, the extracted phrases were validated using syntactic patterns over POS tags. On a limited test dataset, KEP achieved 100% precision for all three target relation types and recall rates of 24%, 33% and 66% for exemplification, definition and partition relations respectively (Bowden et al., 1996).

After the information about the target has been provided, linguistic patterns and templates can be acquired automatically by constructing the left-hand side of the rule and the target structures, respectively. This approach does not presuppose an a priori defined target structure. Instead, the target structure is constructed dynamically depending on the text content. Nobata and Sekine (1999) employed this approach for automatic pattern acquisition and Riloff and Schmelzenbach (1998) also used it to create templates automatically.

Nobata and Sekine (1999) developed a system for automatic pattern acquisition for Japanese. After collecting relevant articles by keywords, then annotated POS tags and named entity (NE) tags were used to identify initial pattern of each sentence. The pattern is represented by an ordered sequence of lexical features and NE tags. They applied primitive merging techniques to cluster various features on the same position: **(A B C) +(A D C) = A (B | D) C** and omitting non-common elements, thus creating a pattern: **A (any features) C**. A pair of most similar patterns is then merged and the original patterns are discarded. They only attempted to localise information on a sentence level rather than classifying or extracting information. Primitive merging techniques limit the level of acceptable pattern generalisation achieved and thus has a negative effect on recall.

Riloff and Schmelzenbach (1998) assumed that the information to be extracted is exclusively comprised of noun phrase (NPs). Therefore, they used heuristics to create linguistic patterns that represent relevant contexts for extracting of a given noun phrase. Firstly, a set of initial patterns was created by extracting every noun phrases from the training text. Then their relevance in the domain was examined. The patterns were scored depending on the number of extracted NPs found in the domain and on their ratio among all extracted NPs. Finally human reviewers selected the best patterns. Patterns that share the same trigger word and matching syntactic

constraints were merged into a single pattern. Such a generalised pattern could be used to extract several slot values and create a multi-slot template as the result of extraction.

Jiang and Zhai (2007) have evaluated a large space of text-based features traditionally used for learned extractors. They began with a set of basic unit features that included words, part-of-speech tags and entity types. They also used features derived from a parser such as syntactic categories and semantic dependencies. Varying sets of features were selected for learning and they closely evaluated the impact of each setting on two state-of-the-art algorithms to extract seven types of relations. Jiang and Zhai (2007) found that the basic unit features were sufficient to achieve state-of-the-art performance and syntactic and semantic features only helped to improve performance slightly.

2.3 Web Information Extraction (WIE)

The Web contains a massive amount of information mainly in natural language text and its quantity keeps growing exponentially to an almost unlimited volume. Web information extraction (WIE) systems are intended to analyse 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), Question-Answering (QA) and Ontology systems. WIE 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 formally written documents.

2.3.1 WIE from Web

Most Web documents are written under no strict supervision and tend to be written informally. The followings are some characteristics of Web documents that affect information extraction:

Informal writing styles Many Web documents are written freely and do not follow strict writing styles used for journalistic text (Collot and Belmore, 1996). As strict writing markers such as Mr., Dr., Inc. and Ltd. which help to detect entities are often absent in informal Web documents, Natural Language Processing (NLP) tools which rely on such indicators are unlikely to work efficiently and may even cause a significant numbers of extraction errors.

Spelling mistakes and incomplete sentences Web documents often contain spelling mistakes and incomplete sentences, which hinder the syntactic analysis of NLP tools and cause extraction errors.

Large amount of newly generated vocabulary Web documents may contain newly generated vocabularies and abbreviations, which are not in formal dictionaries.

As the Web is huge and highly heterogeneous, unsupervised and self-supervised learning methods became an alternative approach for scaling extraction. A number of systems have been developed adopting these approaches. SNOWBALL (Agichtein and Gravano, 2000) iteratively generates extraction patterns using bootstrapping technique and introduces a scalable evaluation method and associated metrics. It takes a small set of seed tuples as inputs, and applies the pattern-entity duality to generate extraction patterns interactively and identifies new relation tuples from unstructured texts. A SNOWBALL pattern is in a form of tuple with five elements <left, tag1, middle, tag2, right> where tag1 and tag2 are Named Entity (NE) tags, and left, middle and right are vectors associating weights with terms. It clusters these tuples using a

simple single-pass clustering algorithm (Frakes and Yates, 1992) with a match function to compute the similarity. Finally, SNOWBALL only collects tuples, which seem correct, based on its confidence measures and adds them to the system as new knowledge. As the target of SNOWBALL is to extract a specific type of relation, the extracted patterns are mainly based on strict keyword matching and it achieved fairly low precision on relations such as 'Merger' and 'Acquisition'.

KNOWITALL (Etzioni et al., 2005) is an unsupervised and domain-independent Web IE system that can handle the heterogeneity of the Web without manually labelled training data. Instead of utilising hand-tagged training data, it selects and labels its own training data, and iteratively bootstraps its learning process. It uses search engines to compute statistical properties and relies heavily on the Web's redundancy. KNOWITALL proposed a novel generate-and-test architecture that consists of two main modules, 'Extractor' and 'Assessor'. During the extraction phase, KNOWITALL employs a set of eight domain-independent extraction patterns, inspired by Hearst (1992) to generate candidate facts. For example, the generic pattern "NP1 such as NPList2" indicates that the head of each noun phrase (NP) in the list NPList2 is a member of the class named in NP1. The Assessor automatically evaluates a probability that each extraction is correct using Pointwise Mutual Information (PMI) statistics, extended from Turney's PMI-IR algorithm (Turney, 2001). Based on the PMI statistics, KNOWITALL associates a probability with every candidate fact extracted that enables it to automatically adjust the trade-off between precision and recall. Its running time increases linearly with the size and number of Web pages it scans. Because KNOWITALL requires domain-specific input such as a set of predicates that indicate its extraction focus, it is a relation-dependent system. Thus, it is necessary to provide a

set of target relations by manual input and it requires a laborious bootstrapping process for each target relation that significantly slows down the system.

The Self-supervised Relation Extraction System (SRES) (Rozenfeld and Feldman, 2008) extracts instances of relations from the Web without human supervision. It is inspired by KNOWITALL and based on the same assumption that the Web corpus is highly redundant. SRES firstly collects sentences from the Web, and then generates seeds using a small set of generic patterns instantiated with the predicate keywords. Next, it uses the seeds to learn likely patterns. With Instance Extractor, it finds all proper and common NPs in sentences using an external shallow parser. In this phase, an NER filter can be used optionally which helped to boost recall by 5~15%. Finally, its classifier assigns a confidence score to each extraction in order to keep appropriate extractions and filter errors.

2.3.2 WIE from Wikipedia

Wikipedia, one of the world's most popular Websites, is a collaborative Web-based resource that is being constantly edited by numerous authors. It has been a popular source for WIE as it contains both comprehensive and high quality information. It is written under less strict supervision than journalistic documents and allows a freer style of writing such as using colloquial expressions. Wikipedia is a hypertext document collection with a rich link structure. Witten (2011) described how to utilise Wikipedia's internal hyperlinks for relational information and their anchor texts as lexical feature to extract semantic relations between concepts.

Usually in each page of Wikipedia, the first sentence holds the definition of the subject entity. Wang et al. (2007b) claimed that the Wikipedia entity features are more powerful than traditional Named Entity Recognition (NER) because they provide more fine-grained descriptions for an

entity. Each article is assigned at least one category. The categories are structured in a directed acyclic graph, which yields a hierarchy of categories. The Infobox, which forms a set of subject-attribute-value triples, provides a general description of the article's subject (an entity). Each triple generally demonstrates a relation between two entities. Thus, Infoboxes are often used as a source of training data, allowing for self-supervised learning. Wikipedia exhibits many of the challenging properties of collaboratively edited data: it contains contradictory data, inconsistent taxonomical conventions, errors, and even spam. It also has a low level of data redundancy as only one article exists for each unique concept. This characteristic increases the need for deep syntactic analysis for extraction compared to recent WIE approaches (Agichtein and Gravano, 2000; Etzioni et al., 2005), which rely on the Web's information redundancy.

Recently, a number of WIE systems have been developed with specific focus on Wikipedia: DBPedia (Auer et al., 2007), work by Ponzetto et al. (2007), YAGO (Suchanek et al., 2007; 2008), SOFIE (Suchanek et al., 2009), KYLIN (Wu and Weld, 2007), KOG (Wu and Weld, 2008) and PORE (Wang et al., 2007a). Most of these systems utilise the semi-structured part of Wikipedia such as infoboxes and the category system. Ponzetto et al. (2007) focus on extracting a large scale taxonomic hierarchy from Wikipedia's categories, while DBpedia and YAGO construct full-fledged ontologies from Wikipedia's infoboxes and the categories. The YAGO system extends WordNet using information extracted from Wikipedia's category tags where leaf category tags are mapped to WordNet nodes with rule-based and heuristic methods. On the other hand, SOFIE can process the unstructured text part of Wikipedia articles. It cannot only handle Wikipedia unstructured text, but also can deal with any other natural language text. The aim of the SOFIE system is to offer a means of automatically extending an ontology by new information extracted while preserving the consistency of the existing ontology. While DBpedia

and YAGO can only extract a limited number of predefined relations from the infoboxes and categories of the Wikipedia, KYLIN extracts any attribute values from both semi-structured and unstructured parts of the Wikipedia. KYLIN aims at filling the infoboxes as Wikipedia's infoboxes suffer from several shortcomings such as incompleteness, inconsistency, schema drift and lack of typing. KOG is the KYLIN Ontology Generator, which constructs a broad and general-purpose ontology using KYLIN's output. KOG unifies different attribute names and derives type signatures. It uses Markov Logic Networks model (Richardson and Domingos, 2006) to jointly infer the WordNet mapping and ISA classification. The WordNet mapping achieved 97% accuracy. KOG applies the idea of joining Wikipedia and WordNet from YAGO and extends it by automatically discovering new relations. It also utilises the idea of taking attributes as relations from DBpedia and improves this by canonicalising the relations. Wang et al. (2007a) have proposed the Positive-Only Relation Extraction (PORE) approach. PORE is a pattern matching approach that has been developed for relation instance extraction from Wikipedia. It utilises entity features to figure out whether a pair of entities is an instance of the target relation or not.

2.4 Open Information Extraction (OIE)

Web Information Extraction (WIE) 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 Information Extraction (OIE) differs from traditional IE in that its goal is to require no pre-defined target relations and no extraction models for individual target relations. It aims to extract tuples representing all possible relationships

among entities in a given corpus. OIE is intended to reduce the amount of time necessary to find desired information from the Web. The Open IE task focuses on identifying both the entities E1 and E2, which are believed to participate in a relation R, and the salient textual cues that provide evidence of the relation between them and to extract the tuple (E1, R, E2).

Shinyama and Sekine (2006) developed a Pre-emptive IE framework in order to avoid relation specificity using an unsupervised extraction process and unrestricted relation discovery. Firstly, the system clusters documents using pairwise vector space clustering. Within each cluster, the system conducts named entity recognition, reference resolution and linguistic parsing, and utilises the outputs to form relational patterns, which are used in a 'meta-clustering' stage. Ideally, these clustering processes aim to divide the given corpus into sets of documents assumed to include entities participating in similar relationships. Sekine (2006) proposed a new paradigm "On-Demand Information Extraction (ODIE)" in order to eliminate the high customisation cost from target domain changes. An ODIE system applies pattern discovery, paraphrase discovery, and extended named entity tagging to automatically discover patterns and extract information on new topics of interest. As these systems relied highly on given documents and entity clustering, they were unlikely to meet the scalability requirement necessary for Web IE.

Open Information Extraction (OIE) systems have performed successfully on massive, open-domain corpora from the Web and Wikipedia (Banko et al., 2007; Zhu et al., 2009; Wu and Weld, 2010; Fader et al., 2011). The extractions from OIE systems have been used to support tasks such as acquiring common sense knowledge (Lin et al., 2010), learning selectional preferences (Ritter et al., 2010), recognising entailment (Schoenmacker et al., 2010; Berant et al., 2011) and mapped onto existing ontologies (Soderland et al., 2010). Recent OIE systems for Web-scale IE such as TEXTRUNNER (Banko et al., 2007; Banko and Etzioni, 2008),

StatSnowball (Zhu et al., 2009), WOE (Wu and Weld, 2010) and REVERB (Fader et al., 2011) are described in following sections 2.4.1 to 2.4.4, respectively.

2.4.1 TEXTRUNNER

TEXTRUNNER (Banko et al., 2007; Banko and Etzioni, 2008) is the first Open Information Extraction (OIE) system for WIE. Two versions called O-NB and O-CRF respectively have been constructed. The O-NB system treated the OIE task as a classification problem using a Naïve Bayes classifier (Banko et al., 2007). The recent O-CRF system, on the other hand, treated the OIE task as a sequential labelling problem using 'Conditional Random Fields (CRF)' classifier (Banko and Etzioni, 2008). The O-CRF system outperforms the O-NB system almost doubling recall.

Similar to KNOWITALL (Etzioni et al., 2005), TEXTRUNNER consists of two modules, extraction and assessment modules, with a separate phase for training the extractor. The Single-Pass Extractor module scans through the entire corpus once, processing each sentence separately and efficiently, and outputs extraction tuples. Its only input is the given corpus. The Assessor module takes the set of extraction tuples to evaluate whether each tuple is correct based on its probability. It only requires the extracted tuples as input, unlike KNOWITALL which requires search engine counts. It also does not need manually labelled examples like traditional information extraction systems. The Self-Supervised Classifier trains itself using automatically labelled examples from an unlabelled formally written corpus, an unlexicalised parser (Klein and Manning, 2003), and a set of heuristic rules.

The Single-Pass Extractor scans over the input documents and processes each sentence in three steps. Firstly it annotates each sentence with part-of-speech (POS) and noun phrase (NP)

chunks using lightweight and maximum-entropy models (Ratnaparkhi, 1998) from the OpenNLP toolkit. It avoids the direct use of a parser on each sentence because parsers are not very robust on informally written Web sentences and using a parser can also delay processing time. Instead, TEXTRUNNER utilises the features from a parser during the training of the Self-Supervised Classifier with a formally written corpus. Secondly, the extractor identifies candidate tuples. For each pair of noun phrases within the pre-set distance and subject to several other constraints the extractor generates a candidate tuple treating the pair of noun phrase as entities and the text between them as the relation string. The relation string is heuristically cleaned up by eliminating stopwords and non-essential adverbial or prepositional phrases. It then applies the Self-Supervised Classifier to classify whether the candidate tuples contain appropriate relations. If the classifier judges the candidate reliable, the candidate tuple of a binary relation form T = (E1, R, E2) is extracted, where E1 and E2 are entities and R is the relation between entities. In the final step, the extractor analyses the candidate tuples and selects the correct tuples by applying the Self-Supervised Classifier. Then, all selected tuples are sorted and merged, resulting in the final set of tuples, which includes only one copy of every extraction with a count of how often each was extracted.

The Redundancy-Based Assessor, based on the URNS model (Downey et al., 2005), applies global counts to the finally stored extractions to assign each tuple a probability, which is used to rank or filter the extractions. The URNS model is an unsupervised model for evaluating the correctness of extractions based on its redundancy. For example, if a tuple is extracted several times in different sentences, the model assigns it a higher probability than if it is extracted only a few times. The exact probability score depends on the total number of extractions. Downey et al.

(2005) proved that the URNS model assigns far more accurate probabilities than the noisy-or model or PMI-based techniques.

2.4.2 StatSnowball

StatSnowball (Zhu et al., 2009) performs both relation-specific IE and OIE with a bootstrapping technique, which iteratively generates weighted extraction patterns. It employs shallow features only such as part-of-speech tags. The StatSnowball system requires pre-defined entities while the TEXTRUNNER, WOE and REVERB systems do not. The StatSnowball system accepts tuples with two entities as input such as (E1, E2, key relation) for traditional IE, and (E1, E2, ?) for OIE to find relation patterns through bootstrapping. In the StatSnowball system, 30 sentences were randomly selected as initial seeds for bootstrapping and the results were then averaged over 10 runs. Each run could have various bootstrapping iterations e.g. from 4 to 10 (Zhu et al., 2009). In StatSnowball, two different pattern selection methods are introduced: the l1-norm regularised pattern selection and a heuristic-based pattern selection.

2.4.3 WOE

Wu and Weld (2010) proposed a new approach to OIE: the Wikipedia-based Open Extractor (WOE), which uses Wikipedia for self-supervised learning. It generates relation-specific training examples by heuristic matching between Wikipedia Infobox attribute values and corresponding sentences in the Wikipedia article as used in KYLIN (Wu and Weld, 2007) and LUCHS (Hoffmann et al., 2010). Then WOE utilises these examples for relation-independent training to learn an unlexicalised extractor similar to TEXTRUNNER's self-supervised learning. WOE used two types of lexical features: POS tag features and dependency parser features. Thus, it can operate in two modes: a CRF extractor trained with POS tags, WOE*pos*, which runs as quickly as

TEXTRUNNER (Banko and Etzioni, 2008); a pattern classifier learned from dependency parser features, WOE*parse*, which outperformed the WOE*pos* in both precision and recall but runs 30 times slower than TEXTRUNNER. Compared with TEXTRUNNER on three corpora, WOE*pos* achieved F-measures between 18% and 34% higher while WOE*parse* yields F-measures between 72% and 91% higher than TEXTRUNNER. While Jiang and Zhai (2007) concluded that parser-based features are unnecessary for information extraction, Wu and Weld proved that abstract dependency paths are a highly informative feature when performing unlexicalised extraction through the performance of WOE*parse*.

2.4.4 REVERB

Fader et al. (2011) reviewed the problems of the state-of-the-art OIE systems such as TEXTRUNNER (Banko and Etzioni, 2008) and WOE (Wu and Weld, 2010) where system outputs frequently include uninformative and incoherent extractions. In order to resolve these problems, they proposed two simple syntactic and lexical constraints on binary relations expressed by verbs. Moreover, unlike previous OIE systems, REVERB is 'relation first' rather than 'arguments first', which assists in avoiding the common errors of previous OIE systems. REVERB significantly outperforms TEXTRUNNER and WOE in both precision and recall.

REVERB takes sentences annotated with part-of-speech and noun phrase chunks as input and returns a set of binary relation tuples such as (E1, R, E2) like TEXTRUNNER. REVERB consists of three processes: relation extraction, argument extraction and assigning a confidence score. In the relation extraction process, relation phrases that satisfy the syntactic and lexical constraints are identified holistically rather than word-by-word. For each verb v in a sentence, it finds the longest sequence of words rv such that

(1) rv starts at v

(2) rv satisfies the syntactic constraint, '**V | VP | VW*P**' where **V=(verb particle? adv?), W=(noun | adj | adv | pron | det) and P=(prep | particle | inf. marker)**

(3) rv satisfies the lexical constraint that rv exists in relation phrase dictionary which is collected from 500 million Web sentences and contains 1717539 normalised relation phrases.

In the argument extraction process, the pair of noun phrase (NP) arguments that is located nearest to the relation phrase and is not a relative pronoun and WHO-adverb or existential "there", are detected as entities for each identified relation phrase. In the final process, the extractions are assigned a confidence score using a logistic regression classifier, which applies a heuristic confidence function features. These heuristic features are efficiently computable and relation independent. The confidence function is trained by manually labelling the extractions of 1000 Web and Wikipedia sentences as to whether they are correct or not.

REVERB requires only a small set of manually labelled data for the confidence function while previous OIE systems used a large number of heuristically labelled training examples (e.g. 200,000 sentences used for TEXTRUNNER and 300,000 sentences for WOE). Heuristically labelling training examples avoids the effort of manual labelling but results in noisy labels and can give a misleading distribution of positive and negative examples. Furthermore, Fader et al. (2011) claimed that learning a confidence function is a much simpler task than learning a full model of relations using a large set of training examples as is done in TEXTRUNNER or WOE.

2.4.5 Discussion

Most OIE systems are developed using Machine Learning (ML)-based approaches and require a large amount of training data. They generally use self-supervised learning, which generates a

labelled training dataset automatically using some heuristics. For example, TEXTRUNNER uses an NLP tool to label entities and a dependency 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 tool errors since it relies on prior automatic labelling from NLP tools. This seriously affects the system's performance as Banko and Etzioni (2008) noted, for example when a verb is incorrectly tagged as a 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 WIE because of the different characteristics of Web documents as discussed in section 2.3.1.

In order to address these problems, we present RDROIE (Ripple-Down Rules based Open Information Extraction), which supports incremental knowledge acquisition (KA) and efficient knowledge base (KB) maintenance. The benefit of the RDR technique is that the KB is built incrementally while the system is already in use. Therefore, it requires no large annotated training dataset. In the RDROIE system, the user creates rules when the extraction result provided by the system is incorrect. Since rules are generated by humans, the RDROIE system is perhaps more likely to be able to handle NLP tools errors and informally written Web documents. The obvious limitation of the RDROIE system is that it is trained for a particular domain of interest; however, since WIE in practice will be used for a specific application domain there is a trade-off between the speed with which rules can be written for the specific domain and the breadth of the domain. In later chapters we will discuss how this trade-off can be avoided by using a hybrid of RDROIE and general purpose systems such as the REVERB system.

Chapter 3. Ripple Down Rules (RDR)

The Ripple Down Rules (RDR) approach was motivated from the frustrating experience of knowledge engineers during maintenance of a traditional rule-based medical expert system known as GARVAN-ES1 system. Observation of the GARVAN-ES1 experts' behaviour suggested that experts do not provide an explanation of how they reach their judgment, rather they provide a justification of their judgment based on the situation. (Compton and Jansen 1990) Experience with the GARVAN-ES1 system led to the development of the first RDR system known as Pathology Expert Interpretative Reporting System (PEIRS), which was used to add clinical interpretations to chemical pathology laboratory reports. (Compton et al., 1992; Edwards and Compton, 1993). PEIRS went into use with about 200 initial rules and expanded its knowledge base to 2000 rules while in routine use. The rules were added by expert pathologists without the need of knowledge engineers. On average, 2 to 3 rules were added per day, taking

about 15 minutes (Edwards, 1996). Notably, the time required for rule addition was independent of the knowledge base size.

The philosophy underlying RDR distinguishes it from other knowledge engineering approaches. RDR is motivated by a situated cognition view where knowledge is acquired incrementally from experts as they deal with real cases requiring their clinical judgement rather than knowledge being structured according to some model (Compton and Jansen, 1988; Compton and Jansen, 1990). This is essentially a combined rule-based and case-based reasoning approach (Kang and Compton, 1994). The cases provide the context, for which a rule or a correction rule is provided. For example, a new rule is added when the classification from the KB is incorrect for the given case. The new rule represents an explanation for why the classification should be different from the KB's classification for the case at hand.

The RDR approach is different from traditional rule-based approaches in focussing on knowledge maintenance. Traditional rule-based systems pay little attention to Knowledge Base (KB) maintenance as it was expected that thorough upfront domain analysis conducted by a knowledge engineer and a domain expert together would be adequate, and if not it would be easy to add another rule. It is easy to add a rule, but not to debug the rule to be sure it does not cause any unexpected changes. This debugging results in a significant increase in Knowledge Acquisition (KA) time as the KB grows because of the need to ensure no inconsistencies are introduced in other parts of the rule tree. On the other hand, RDR systems build a KB incrementally using the domain expert's justification without a knowledge engineer, while the system is in routine use. The expert provides a justification why the classification should be different from the system's classification result for the given case and creates exception rules to repair the classification result. The position of the new rule will be determined automatically and

the consistency of the new rule will be checked automatically before being added to the KB. Consequently, the time required for KA in RDR system is independent of the KB size. Easy maintenance and validation are main reasons that RDR has achieved commercial success. For example, in one report of a particular laboratory, the Labwizard commercial RDR system processed about 30 million patient reports using about 14 knowledge bases. Labwizard, took on average about 1 minute of expert time to add a rule and debug the knowledge base assessed across thousands of rules (Compton et al., 2006).

Incremental knowledge acquisition using RDR has a number of strengths over automated knowledge acquisition using Machine Learning (ML). Supervised ML algorithms require cases to be classified and labelled upfront to build a training dataset. In many domains, this process is done manually and is prone to error and inconsistency simply due to human fatigue, memory limitations, emotions and so on (Wang et al., 1996). On the other hand, RDR's incremental approach assumes that there will be always errors and it concentrates on fixing the errors. RDR requires experts to check each case's classification against their previous judgment and justify why this new case deserves a different classification. With RDR, an error can be corrected manually the first time it occurs with an RDR system but with machine learning, one needs to keep monitoring cases until sufficient examples of each type of errors have been accumulated for the learning system to learn correctly.

3.1 Single Classification RDR (SCRDR)

A Single Classification Ripple Down Rules (SCRDR) tree is a finite binary tree with two child nodes. The edges connecting the child nodes are typically called *except* and *if-not* (or *false*)

edges. Each node in a tree represents a rule. A rule consists of condition part and conclusion part: IF condition THEN conclusion. In SCRDR, the inference engine starts at the root node of the tree and evaluates whether the given case satisfies the corresponding rule N's condition. If the rule N is fired, then the case is passed on to the *except* child of N if it exists. Otherwise, the case is passed on to *if-not* child of N if it exists. The last fired rule is the conclusion given and SCRDR only allows one conclusion. To ensure that a conclusion is always given, the condition of the root node (called the default node) is always satisfied and returns a NULL classification conclusion.

A new rule is added to an SCRDR tree when the Knowledge Base (KB) returns incorrect or NULL classification result. The new rule is attached to the last node in the inference path as an *except* child if the last node is the fired rule. Otherwise, it is attached as an *if-not* (or *false*) child. The case that triggered the creation of the new rule R is stored together with the rule and called the cornerstone case of a rule R. It is retrieved when an exception rule Re of a rule R needs to be added into the KB. The cornerstone case is intended to assist the expert to form a valid child rule Re of a rule R by comparing differences between the cornerstone case of R and the case at hand which triggered the addition of new rule Re.

Every time a new exception rule is entered into a KB, a difference from the previous rule has to be identified by the expert. RDR-based systems effectively store all previously seen cases in association with the rules, which correctly classified those cases. Whenever an exception child rule Re is added under a parent rule R, the rule Re is checked to not 'fire' for the cases stored under the rule R. That is, the exception rule Re must be satisfied by the case at hand and not satisfied by all cases stored under the rule R.

RDR-based systems usually use attribute and value pairs to form a condition of a rule (Compton and Jansen, 1990; Compton et al, 1993; Shiraz and Sammut, 1997). To create an exception rule Re, experts only need to choose conditions from the 'difference list' (Compton and Jansen, 1990) which includes all attribute and value pairs satisfied by the case at hand which triggered the creation of Re and excludes all attribute value pairs satisfied by the stored cases under R. However, it is not always practical to build the 'difference list' because some combinations of differences may result in a list that is very long. For example, when the attributes are multi-valued or continuous, the 'difference list' becomes complex and does not provide efficient assistance for the expert. As RDR-based systems focus on how to validate a KB with cases comparing the case at hand with all associated cases with R, difference lists are not critical (Richard, 2009).

3.1.1 Discussion

SCRDR was applied to learn complex control knowledge where it was combined with machine learning techniques to capture an expert's knowledge used in flying and landing a simulated single engine plane (Shiraz, 1998). The high-level control knowledge was obtained using an RDR knowledge base while the sub-cognitive skills were acquired using a machine learning component in the system. It has been also used in image processing (Kerr and Compton, 2002), multi-agent coaching in soccer simulation (Finlayson and Compton, 2004) and planning (Kwok, 2002).

SCRDR was successfully used in the medical expert system PEIRS which was used to add clinical interpretations to pathology reports (Compton et al 1992; Compton et al 1993). An issue arising from PEIRS is that a patient may have more than one disease. PEIRS handles this by treating multiple independent diseases as compound diseases. However, this approach could

exponentially increase the number of knowledge acquisition sessions as the number of combinatorial classes proliferates (Kang et al., 1995). An alternative solution would be to separate domains and build independent multiple SCRDR KBs for each domain. This approach has been used by Shiraz (1998) and Pham and Hoffmann (2002). However, in many domains it is hard to separate sub-domains and a clumsy work-around would be necessary. The Multiple Classification Ripple Down Rules (MCRDR) methodology was developed to address this problem (Kang et al., 1995). While SCRDR just allows a single refinement to a rule, MCRDR allows multiple refinements supporting multiple classifications for a case.

3.2 Multiple Classification RDR (MCRDR)

A Multiple Classification Ripple Down Rules (MCRDR) is an 'n-ary' tree structure with only true (*except*) edges. A case is evaluated by passing it to the root node, which is always satisfied. MCRDR evaluates all the rules at the first level of the KB (child nodes of the root node). When a rule is satisfied, all children of that node are tested recursively where the children's conclusions overwrite the parent's conclusion. The inference process stops when there are no more children nodes to evaluate and it ends up with multiple paths resulting in multiple conclusions. The last true node at the end of each pathway returns a set of classification results. MCRDR introduced the idea of stopping rules that make no conclusion, or rather return a NULL classification in order to prevent wrong classifications by over- generalized, or specified rules. (Kang et al., 1995)

MCRDR may have multiple cornerstone cases for a rule R that must be distinguished from the case that triggers a new rule Re. Every time when a new rule Re is added under a parent rule R, Re should not be fired on cases that are already correctly handled by its parent rule as well as its

sibling rules excluding cases which include the same classification. Usually this process is assisted through the user interface by presenting one cornerstone case at a time. The expert can refine the condition of Re until all cornerstone cases of R and R's children rules are not fired. Kang et al., (1995) have shown that a sufficiently precise rule is created usually after seeing two to three cornerstone cases. The expert can make effective conditions quickly after reviewing a few examples. Thus, inspection of multiple cornerstone cases is not a major problem in practice.

3.2.1 Discussion

A numbers of studies have shown that MCRDR can produce more compact and less redundant KB more quickly than SCRDR even for single classification domains (Kang et al., 1995; Kang, 1996; Kang et al., 1998). MCRDR also needs fewer cases to be entered to cover multiple subdomains than SCRDR (Richards, 2009). In MCRDR it is possible to utilise different attributes of the case to create different rules and conclusions for each area of focus or subdomain (Richards, 2009).

MCRDR was employed successfully in configuration tasks to provide advice for Ion Chromatography methods (Ramadan et al., 1998; Compton et al., 1998). MCRDR was also used to solve a benchmark room allocation problem known as Sisyphus I (Richards and Compton, 1999). For this task, MCRDR was modified to incorporate backtracking whilst in search of a solution. MCRDR is the basis of the widely used commercial system referred to in the paper showing one or two minutes required per rule construction (Compton et al 2006, Compton et al 2011).

MCRDR may need to refine more than one rule to correct the same conclusion because it is likely that there are multiple true pathways. In order to reduce the repeated KA effort when

adding numbers of exception rules to refine same classification result automated patching is supported in commercial implementations (Compton et al, 2006). It allows the expert to define the first patch then the system automatically adds exception rules to other rules to correct the same misclassification. Notwithstanding that the repeated KA effort seems inefficient, empirical and simulation studies have proven that the depth of correction is usually two or three rules (Kang et al., 1995). Thus, in practice the repetition is not a critical problem for system performance.

3.3 Generalised RDR

The success of basic RDR strategies raised the question of whether a general strategy of incremental KA is possible, which could be employed for a wide range of problem types and generalised RDR strategy have been proposed (Compton et al, 2004; Cao and Compton, 2005). Generalised RDR aims to solve the classification task over multiple inference cycle, where the solution has a finite number of components and relations between components without the need for searching, backtracking or changing parts of the solution already developed. In practice, an expert proposes a solution and then revises it when they find problems. Generalised RDR was motivated based on the assumption that any backtracking in reasoning implies an error and shortcoming of the expert.

In generalised RDR, a piece of output is added to a case, then the inference process rerun with the augmented case. If the expert decides the derived conclusion (output) as incorrect, they provide a correction rule in the normal way. As usual in RDR the expert must select sufficiently precise conditions for the rule to exclude any stored cases from firing the rule. The case that

triggered the addition of new rule is stored as a cornerstone case. It will become a cornerstone case for all rules that satisfied on this case in building the final output. The inference process is repeated until the case passes through the knowledge base system with no more output being added.

With RDR the expert is likely to deal with individual errors, but accumulative refinement should support the system in building a very high level of expertise. Though this seems a rather conventional inference structure it has significant differences from conventional inference. It does not require conflict resolution strategy. Rules are processed in an exact order and once the final conclusion on any refinement path is reached it is instantly added to the case and cannot be altered at a later stage of inference. The only way an addition to the case can be altered is by an expert adding a refinement rule for the rule that derived the conclusion. Importantly, a rule with the same (or an alternative) conclusion as a previous rule that has been fired on is not considered to fire and nor are its children evaluated.

3.3.1 Repeat-Inference to MCRDR (RIMCRDR)

The generalised RDR approach evolved from an earlier development of Repeat Inference MCRDR (RIMCRDR) approach whereby the existing MCRDR method was enhanced to use classifications (output) of the system as rule conditions (Compton and Richard, 1999; Compton and Richard, 2000). RIMCRDR keeps consistency by performing the inference in strict chronological order of the rules added and not allowing retractions (Compton and Richard, 2000). This approach, however, included many constraints with regard to when and how these types of rules could be used, and how the knowledge base must be inferred and interpreted. These constraints are added to exclude potential cyclic rules – rules which depend on the existence of a classification, yet upon firing, may trigger that the same classification be retracted. Historically

RIMCRDR was developed specifically for a configuration and then resource allocation problem and generalised RDR is in fact a more general statement of the RIMCRDR method used for these problems.

3.3.2 Linked Production Rule

With SCRDR and MCRDR the inference engine provides little control of inference as the sequence of rules evaluated is determined by the exception structures used. With RIMCRDR the inference engine does provide some control of the inference sequence similar to conventional rule based systems. To minimise the maintenance problems that might arise it was desirable to remove control of the inference sequence from the inference engine.

It was realised that the SCRDR, MCRDR and then RIMCRDR could be represented by linked production rules (Compton et al., 2011). In linked production rules, each evaluated rule guides the next step of inference and the inference engine does not require meta-heuristics or conflict resolution strategy. The expert is only needed to provide domain knowledge while the links of inference steps are added automatically. Linked production rules is comprised of following three components: [condition], [case action-list] and [inference action-list]. Inference actions do not modify the case, but specify which rule is to be evaluated next. Case actions modify the case in the conventional way. Inference actions applied every time when the rule is satisfied or failed. This gives the following structure: If [condition] then [case action-list], [inference action-list] Else [inference action-list]. It is shown in (Compton et al 2011) that generalised RDR can be represented by linked production rules with all control of inference being determined by the links.

3.4 RDR in NLP Tasks

In our project, we will need to write rules about text so the following is a brief discussion of various uses of RDR for Web text. Many of the Web-based approaches have moved RDR into document retrieval, information extraction, natural language processing and text summarisation.

A number of RDR-based approaches have been developed to support document retrieval from the Web. Kang et al (1997) combined key word searching with CBR indexing to provide a filter and guide searching. Kim et al (1999) developed a prototype to support document classification by building a KB of words or phrases that appeared in the document (rule conditions) linked to documents (conclusions). They used keywords to identify documents of interest, but they were then classified them using words that appeared in the documents. Kim and Kang (2008) introduced personalized search query generation methods reusing MCRDR document classification knowledge. Park et al. (2004a, 2004b) focused on the information management system of context changes in Web document classification. They applied the MCRDR method to successfully manage context changes and maintain information classification knowledge. Their system also achieved rapid system performance recovery with only a small number of additional cases.

There has been similar RDR research focused on email management. Ho et al. (2003) proposed EMMA (E-Mail Management Assistant) for an email management assistant based on RDR. It includes message sorting into virtual folders, prioritising, reading, replying, archiving and deleting emails. EMMA used MCRDR for classification but also adopted a machine learning technique (naïve Bayesian algorithm) in three alternative ways to enhance coverage and usability by suggesting keywords to assist the user define rules. EMMA achieved accuracy between 95% and 99% during the trial period. Wobcke and Krzywicki (2008) reported an evaluation of EMMA on 16998 pre-classified messages to demonstrate the performance of EMMA on large-

scale dataset in realistic organisational contexts. EMMA achieved the agreed success criteria for the evaluation and outperformed standard machine learning methods. Krzywicki and Wobcke (2009) introduced a number of methods for e-mail categorisation and keyword rating based on Simple Term Statistics (STS) updated incrementally. They employed the STS on two tasks; one is to predict folders for e-mail classification when many messages are required to remain unclassified. The other is to assist users who define rule bases for the same classification task, by suggesting appropriate keywords for RDR KB construction. For both tasks, the STS method achieved a higher level of accuracy than other machine learning methods when taking computation time into account.

The RDR approach has also been applied to an information extraction task. For example, Pham et al. developed KAFTIE (Knowledge Acquisition Framework for Text Classification), an incremental knowledge acquisition framework to extract positive attributions from scientific papers (Pham and Hoffmann, 2004a and 2004b) to reduce some of the information overload for scientists. KAFTIE used an annotation-based rule language including a customisable shallow parser to specify the rule conditions. In addition, Pham and Hoffmann (2005, 2006) extracted temporal relations that outperformed machine learning algorithms such as C4.5, Naïve Bayes and SVM.

Hoffmann and Pham (2003) also introduced KAFDIS (Knowledge Acquisition Framework for DIScourse processing) for text summarisation task using RDR. This system aims to construct reliable discourse structure graph and the level-of-detail tree based on cue phrases.

The RDR techniques have been used for classic Natural Langue Processing (NLP) tasks, especially for Part-Of-Speech (POS) task. Xu and Hoffmann (2010) proposed RDR Case

Explorer (RDRCE) for the combination of machine learning approach and knowledge acquisition (rule-based) approach in order to improve NLP problems such as Part-Of-Speech (POS). This automatically generated an initial RDR tree by using transformation based learning, but then allowed for corrections to be made. Therefore, RDRCE necessitates labelled training dataset for automatic initial tree creation. They applied RDRCE to POS tagging and achieved a slight improvement over the state-of-the-art POS tagging after 60 hours of KA. Nguyen et al. (2011) introduced a failure-driven approach to automatically restructure transformation rules in the form of a SCRDR tree for POS task. Their approach allows controlled interactions between rules where a rule only changes the results of a limited number of other rules. They achieved the best accuracy published to date of 97.095% on the Penn Treebank corpus and the best performance for Vietnamese VietTreeBank corpus; an accuracy of 92.24% for an open dictionary assumption and an accuracy of 94.61% for close-dictionary assumption.

The Web has brought new challenges and opportunities for employing RDR on various classification tasks. The challenges are usually about how to utilise efficiently the large amount of resources available on the Web. Because the volume of Web documents continually increases and their contents dynamically change, continuously fixing errors via incremental learning may be very useful. As various RDR used NLP systems have been successful, this suggested that we could use RDR for open information extraction.

3.5 RDR and Data Quality

In the real world, data often contains various levels of noise. Poor data quality is a serious problem affecting enterprise data management. Data quality improvement usually requires an

iterative process mainly for writing a set of rules. RDR's incremental learning has been applied successfully to handle various data quality improvement tasks.

Dani et al., (2010) proposed an RDR framework for cleansing noisy street addresses. They demonstrated that RDR is an effective for the address standardisation task as it handles the large amount of variability well using RDR's exception structure. They compared the RDR approach with a conditional random field (CRF) street address standardisation system and an existing commercial tool. Prasad et al. (2011) introduced a data quality improvement tool to assist the data quality practitioner by providing the characteristics of the entities present in the data. They combined their tool with the standard data cleansing workflow and demonstrated how a synonym finder, variant finder and the RDR framework complement each other at the various stages of the workflow. They employed a proactive approach to identify possible unhandled terms and the associated context that can be utilised to handle them upfront.

As discussed above, applying the RDR method to improve data quality is very promising approach. This suggests that, we could also employ RDR method intensively to handle the Web's informality for open information extraction.

3.6 RDR in Conjunction with a General System

Ho et al. (2009) proposed an application combining a general-method and RDR to the problem of classifying pairs of potential duplicate invoices. They demonstrated how to build a prototype on top of SAPs duplicate invoice detection system. RDR rules were added to correct false positive identification of duplicate invoices by the general method. The false positives are

because of highly contextualised information that the general method cannot deal with. This produced extremely good results, removing nearly all the false positive warnings about possible duplicate invoices, with very few rules.

From Ho et al.'s successful result when combining a general method with RDR, we hypothesised that our Hybrid RDROIE system and RDRNER system could improve the result of the REVERB OIE system and the Stanford NER system as end users could add contextualised rules for particular domains, on top of a general system, to fix NLP errors as they were identified.

Chapter 4. Ripple-Down Rules based Open Information Extraction (RDROIE)

In this chapter, we describe the RDROIE (Ripple-Down Rules based Open Information Extraction) system (Kim et al., 2011) that supports incremental Knowledge Acquisition (KA) and efficient Knowledge Base (KB) maintenance. It is a new approach to Open Information Extraction (OIE) task that uses Multiple Classification Ripple-Down Rules (MCRDR) (Kang et al., 1995)' incremental learning technique. The benefit of the RDR technique is that the KB is built incrementally while the system is already in use. Therefore, it requires no large training dataset with annotation. In the RDROIE system, the user creates rules when the extraction result provided by the system is incorrect. Since rules are generated by humans, the RDROIE system is more likely to be able to handle NLP tool errors and informally written Web document.

The RDROIE system aims to extract binary relations in the form of a tuple (ENTITY1, RELATION, ENTITY2) from the informal Web documents. For example, it extracts a tuple

(**Google**, **acquire**, **YouTube**) from a sentence '*Finally, Google acquired the Video sharing company YouTube !*'.

Section 4.1 begins with a detailed description of the RDROIE system architecture and Section 4.2 presents the description of RDR rule and examples of MCRDR structure in the RDROIE system. Section 4.3 demonstrates the process of the RDROIE system through screenshots. Section 4.4 states the experimental setup to evaluate the RDROIE system and section 4.5 describes the initial KB construction of the RDROIE system. Section 4.6 presents a performance comparison and the advantages of the RDROIE system for the OIE task. Section 4.7 provides some discussion of these results.

4.1 RDROIE System Architecture

The RDROIE system shown in Figure 4.1, consists of three main components: preprocessor, tuple extractor and RDR KB learner.

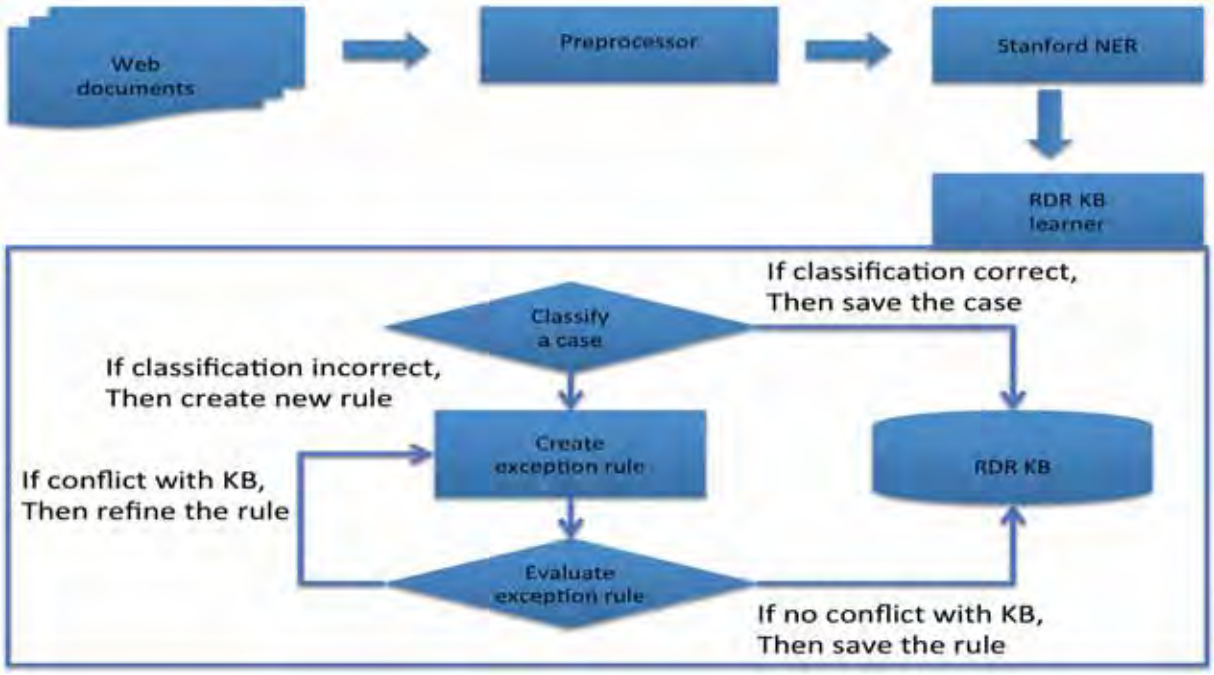

Figure 4.1. Architecture of the RDROIE System

4.1.1 Preprocessor

The preprocessor converts the raw text into a sequence of sentences, and annotates each token for Part-Of-Speech (POS) and phrase chunk using OpenNLP[1]. It also annotates Named Entities (NE) using the Stanford NER[2]. The resulting annotated data is fed into the Tuple extractor.

4.1.2 Tuple Extractor

The tuple extractor contains two entity identifiers. One version identifies entities by named entities annotated using the Stanford NER and the other version detects entities by noun phrase chunk annotated using the OpenNLP. Once entities are identified in a sentence, the tuple extractor generates candidate binary relation tuples. A tuple consists of two entities and the

[1] Available at 'http://incubator.apache.org/opennlp/'
[2] Available at 'http://nlp.stanford.edu/software/CRF-NER.shtml'

relational text between entities (entity1, relational text, entity2). The maximum number of tokens allowed for relational text is seven tokens as a default that can be reset by a user.

For example, in the following sentence two entities are identified as named entities.

"*Never even saw this coming, but apparently Adobe/**ORG** is about to buy Macromedia/**ORG** .*"

(where ORG stands for organisation named entity)

With two entities tagged, the tuple extractor extracts one candidate tuple:

(**Adobe**, **is about to buy**, **Macromedia**)

In the following sentence three entities are identified by noun phrase chunk.

"*Never even saw this/**B-NP** coming/**I-NP** , but apparently Adobe/***B-NP*** is about to buy Macromedia/**B-NP** .*"

(Where B-NP indicates the starting token of the noun phrase and I-NP indicates continuing token of noun phrase)

With three entities tagged, tuple extractor extracts three candidate tuples:

(**this coming,** , **but apparently** , **Adobe**)

(**Adobe** , **is about to buy** , **Macromedia**)

(**this coming,** , **but apparently Adobe is about to buy** , **Macromedia**)

4.1.3 RDR KB Learner

The RDROIE KB system is built incrementally while the system is in use. In the RDROIE system, the user runs the system on a part of the training dataset, gets the classification results

and adds a rule when the classification result is incorrect. The user interface for creating a new rule is shown in section 4.3, and displays not only the rule conditions but also the feature differences between the current case and any stored cases with different classifications but which are also satisfied by the new rule.

Step1: RDROIE classification

For a given case, the system returns the classification result from fired rules in the KB. The conclusion may be correct, incorrect or missing (NULL). If the user decides the classification result is correct, the current case is simply saved under the fired rule in KB.

Step2: Create RDR rule

If the returned classification is incorrect or NULL, the user creates a new rule to be added by selecting case's features identified in the user interface.

Step3: Evaluate and refine RDR rule

Once the new rule is created, the system automatically evaluates all the stored cornerstone cases, which may fire the new rule. The number of cases evaluated will vary depending if the rule is a new rule or an exception rule. If the rule fires one or more of these cases, the user can select differentiating features from the NLP features displayed in the user interface and make the rule more precise. The user may also decide that the previous classification for one or more of the previous cases was incorrect and that the new rule should apply.

4.2 RDR Rule Description

An RDR rule comprises of a condition part and a conclusion part. It has the form:

if CONDITION

then CONCLUSION

where CONDITION indicates more than one condition.

4.2.1 Condition

A CONDITION consists of three components: ATTRIBUTE, OPERATOR and VALUE.

4.2.1.1 Attribute

The system divides a given sentence into five elements in the form of [ENTITIY1BEFORE, ENTITY1, RELATION, ENTITY2, ENTITIY2AFTER]. ATTRIBUTE refers to one of the five elements which the user selected. As mentioned in section 4.1.2, each candidate tuple forms (ENTITY1, RELATION, ENTITY2) and the remaining tokens before or after the ENTITY1 and ENTITY2 elements in the given sentence form the ENTITIY1BEFORE and ENTITIY2AFTER elements. For example, in the sentence,

> "*Never even saw this coming, but apparently Adobe/**ENTITY1** is about to buy Macromedia/**ENTITY2** .*"

The five attribute elements are as follows:

ENTITY1BEFORE: '**Never even saw this coming, but apparently**'

ENTITY1: '**Adobe'**

RELATION: '**is about to buy**'

ENTITY2: '**Macromedia**'

ENTITY2AFTER: ‘**.**’

During the new rule creation, the user can choose any of five elements as an ATTRIBUTE for a rule condition. As the user can utilise not only a candidate tuple (ENTITY1, RELATION, ENTITY2) extracted but also ENTITY1BEORE and ENTITY2AFTER, the rule condition can be set without limitation in the context window. That is, the user can use any features within the sentence rather than only within the candidate tuple extracted.

4.2.1.2 Operator

The RDROIE system provides 5 types of simple OPERATORs as follows:

- hasToken: whether a certain token match exists within the chosen attribute
- hasPOS: whether a certain POS match exists within the chosen attribute
- hasChunk: whether a certain noun/verb phrase chunk match exists within the chosen attribute
- hasGap: skip a certain number of tokens or spaces to generate an extraction pattern
- notHasPOS: whether a certain POS match does not exist within the chosen attribute

In the RDROIE system, the VALUE of the ‘hasToken’ OPERATOR is transformed to its normalised form when the condition is evaluated.

4.2.1.3 Value

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

> “*Never even saw this coming, but apparently Adobe/**ENTITY1** is about to buy Macromedia/**ENTITY2** .*”

If ATTRIBUTE is set to RELATION element which includes ‘**is about to buy**’ and

If OPERATOR is set to ‘hasPOS’

Then VALUE such as ‘VBZ IN TO VB’[3] will be displayed automatically in the user interface.

Once the ‘hasToken’ OPERATOR is selected, the top 10 Synonyms derived from the WordNet are automatically listed in the user interface when the user clicks the ‘show synonyms’ button. Users can deselect synonyms that are too general such as ‘get’, ‘have’, any erroneous or inappropriate synonyms. Selecting synonyms allows the rule to apply to these synonyms as well as the original token.

4.2.1.4 Type of Conditions

Each condition detects the value from the given attribute element. For example, the condition ‘if (RELATION hasToken ‘**buy**’)’ detects the value ‘buy’ from the RELATION element. Conditions are connected with an ‘and’ operation.

There are two types of condition: non-sequence condition and sequence condition. A non-sequence condition applies each condition one by one independently without order. On the other hand, a sequence condition detects a group of words in a sequence order, so a pattern can be detected. A ‘SEQ’ keyword is used in the condition to indicate it as a sequence condition.

For example, a sequence condition:

‘SEQ((RELATION hasPOS ‘**TO**’) & (RELATION hasPOS ‘**VB**’))’

detects whether a ‘TO VB’[4] pattern exists in the RELATION element.

[3] See Appendix C for part-of-speech tags
[4] See Appendix C for part-of-speech tags

4.2.2 Conclusion

The CONCLUSION part gives the classification result for the given case. The RDROIE system handles three different classification tasks: ‘relation existence detection’, ‘relation taxonomy type detection’, and ‘specific relation keyword detection’. The relation existence detection task detects whether a semantic relation exists between the given entity pair, and the relation taxonomy type detection task classifies whether the four taxonomy types of binary relation exist as identified in (Banko and Etzioni, 2008): ‘Verb’, ‘Noun+Prep’, ‘Verb+Prep’ and ‘Infinitive’. The specific relation keyword detection task extracts the actual relation keywords. Therefore, the CONCLUSION part has a following form:

```
(relDetection,          --- whether a relation exists within the relational text
 taxonomy,              --- one of following four binary taxonomy type:
                            Verb | Noun+Prep | Verb+Prep | Infinitive
 relationKeyword).      --- specific relation keyword
```

4.2.3 Examples of RDROIE Rules

The RDROIE system is based on Multiple Classification RDR (MCRDR) (Kang et al., 1995). Figure 4.2 demonstrates MCRDR based KB construction as the RDROIE system runs. The rule numbers in figure 4.2 shows the sequence of KB construction from an empty KB (with a default rule R1 whose condition is always true and returns NULL classification) as the system processes the given cases. In the following examples, cases are described as a candidate tuple (ENTITY1, RELATION, ENTITY2) extracted from a sentence by the Tuple Extractor. The

ENTITY1BEFORE and ENTITY2AFTER elements are excluded in the following examples for simplicity.

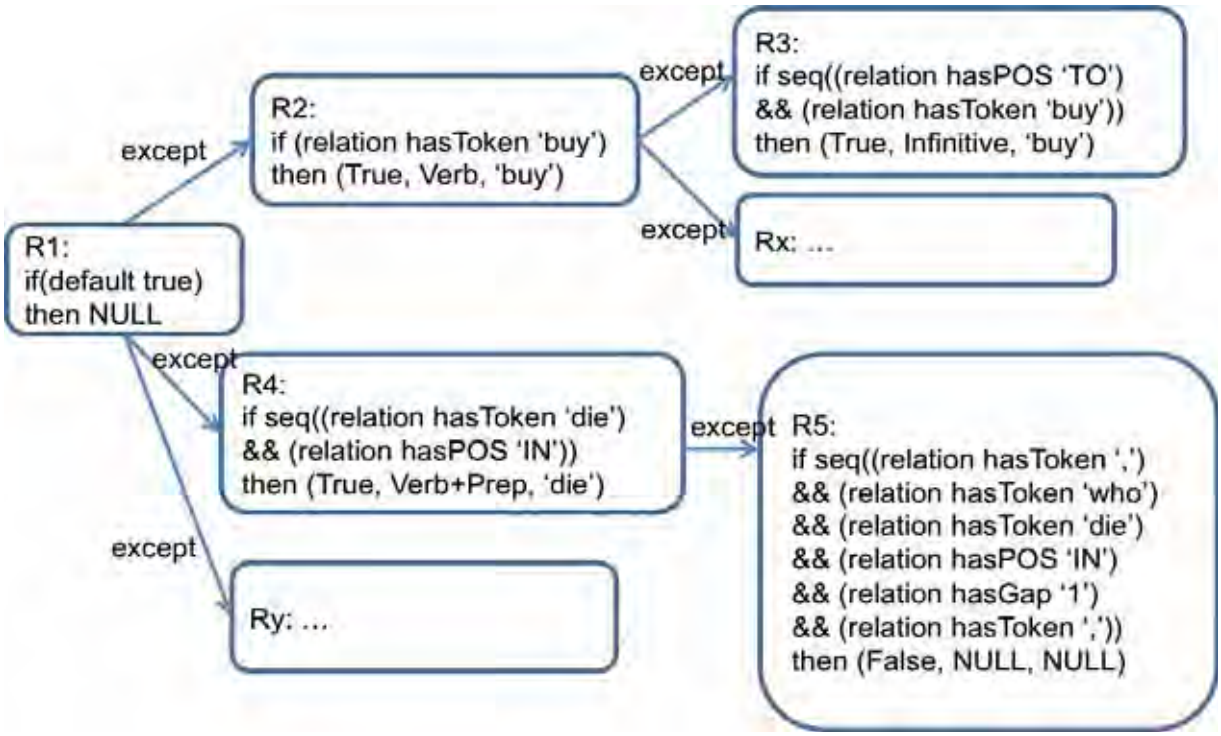

Figure 4.2. MCRDR Structure of the RDROIE System

Case1: A candidate tuple (**Google**, **just bought**, **YouTube**) extracted from a sentence '*Google just bought YouTube for 1.65B.*' is given as a case.

➔ The KB system returns a NULL classification, which is incorrect.

➔ A user generates a new rule R2 under the default rule R1 and specifies relevant synonyms (acquire, purchase) which are also stored in the KB.

Case2: A candidate tuple (**Google**, **to acquire**, **Youtube**) extracted from a sentence '*Google to acquire Youtube for $ 1.65 Billion in Stock at Total Lost in space Says:*' is given as a case.

➔ R2 fires since ‘acquire’ is stored as a synonym of ‘buy’ in the KB, but its relation taxonomy type classification is incorrect.

➔ A user creates an exception rule R3 under its parent rule R2.

Case 3: A candidate tuple (**Google**, **to buy**, **Youtube**) extracted from a sentence ‘*Google to buy Youtube for 1.6 billion.*’ is given as a case.

➔ R3 fires with the correct classification.

➔ A user saves case3 to KB under R3.

Case 4: A candidate tuple (**Mozart**, **died in**, **Vienna**) is extracted from a sentence ‘*Mozart died in Vienna at age 35 of uremia, a kidney disorder that was caused by his excessive consumption of alcohol* .’. is given as a case.

➔ The KB system returns a NULL classification, which is incorrect.

➔ A user creates a new rule R4 under the default rule R1.

Case 5: A candidate tuple (**Charlie Chaplin**, **, who died in 1977 , was born in**, **London**) extracted from a sentence ‘*Charlie Chaplin , who died in 1977 , was born in London to music - hall parents* .’ is given as a case.

➔ R4 fired with the incorrect classification (because ‘died in’ has no semantic relation between entity1 ‘Charlie Chaplin’ and entity2 ‘London’).

➔ A user creates an exception rule R5 under its parent rule R4.

4.3 Knowledge Acquisition (KA) Process for the RDROIE System

In this section, we present a number of screenshots of the RDROIE system's user interface to demonstrate its process while handling three given cases (case 1 to case 3) in section 4.2.3 starting with an empty KB. In the RDROIE system, a user performs the Knowledge Acquisition (KA) process through user-friendly interface, which automatically presents the features needed in each KA step. On the screenshot, each KA step is numbered to show the process order and each process step is described below each screenshot. The RDROIE system is written in Java (Java 1.6) and it uses the OpenNLP system (version 1.5) and the Stanford NER system (version 1.5) to extract NLP features.

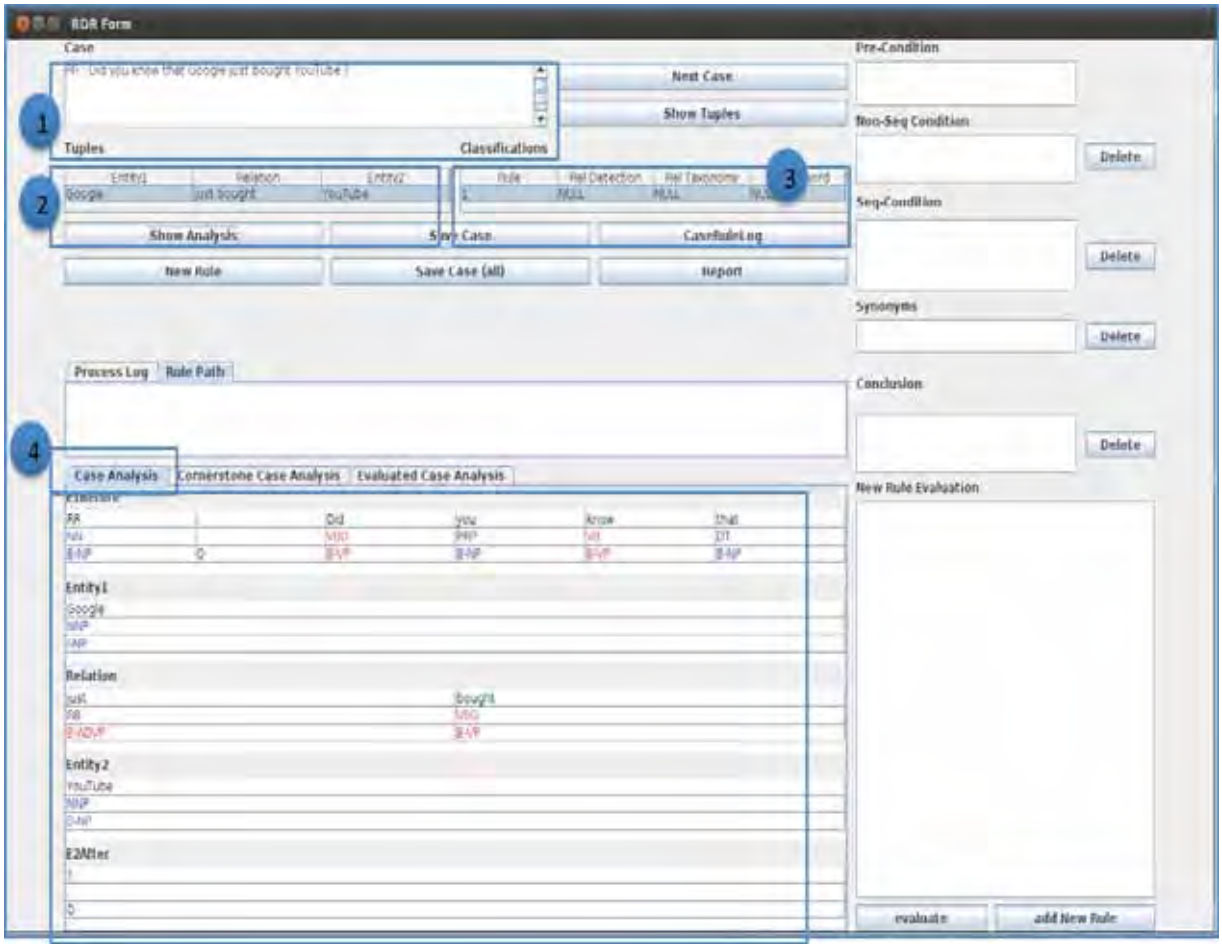

Figure 4.3. The KA process when a new case is entered into the system with empty KB

Step 1: The given sentence '*RR: Did you know that Google just bought YouTube?*' is displayed.

Step 2: The user clicks the 'Show Tuples' button, then a candidate tuple (**Google** , **just bought** , **YouTube**) extracted from the Tuple Extractor is displayed.

Step 3: As the user clicks the extracted candidate tuple and the system automatically returns the classification result (the default rule R1 is fired and returns the NULL classification) from the KB (which is an empty at this stage).

Step 4: The user clicks the ‘Show Analysis’ button and the system presents NLP features such as token, part-of-speech and chunk phrase. The user checks the ‘Case Analysis’ table to figure out the features to make a new rule.

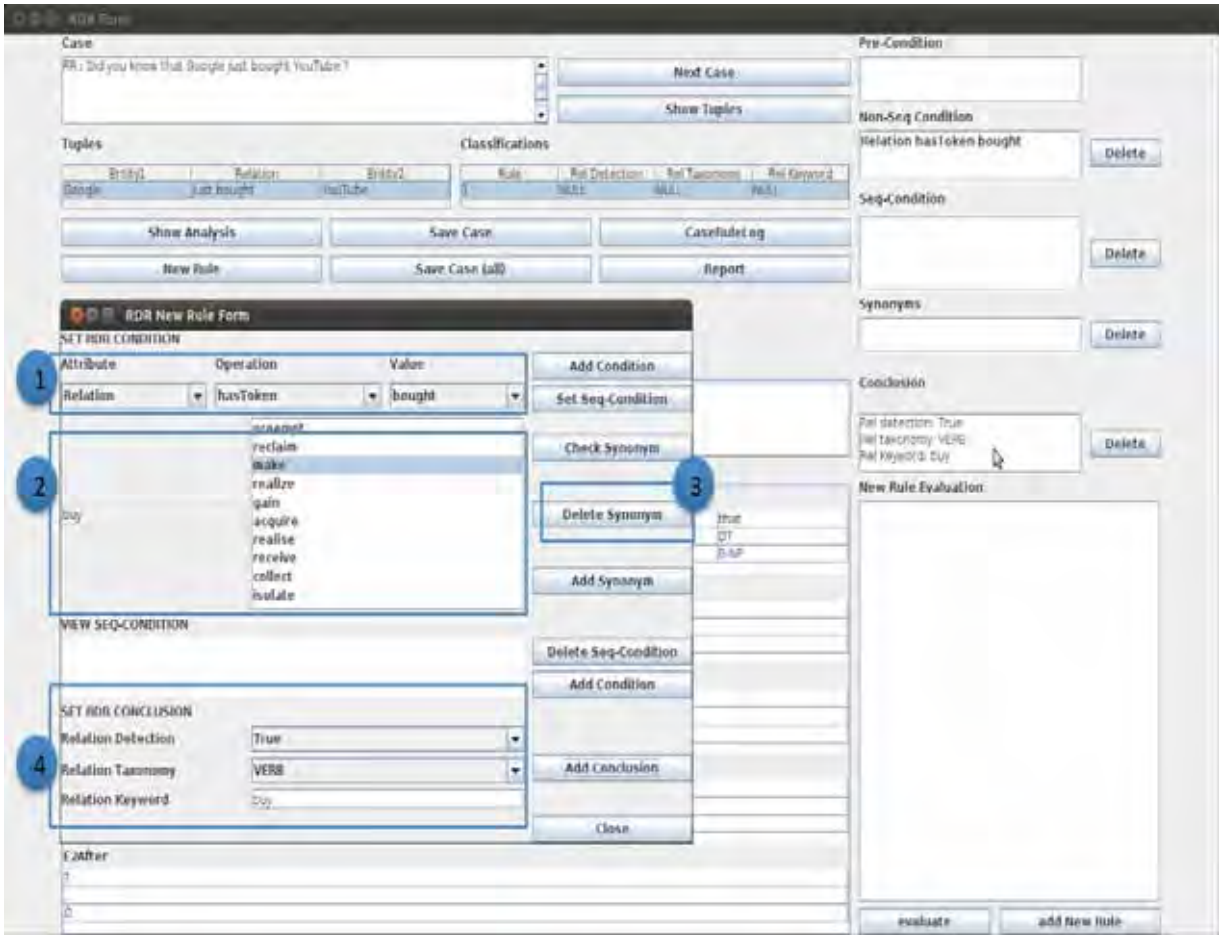

Figure 4.4. The KA process when the user clicks the ‘New Rule’ button to create a new rule and ‘RDR New Rule Form’ window pops up

Step 1: The user selects the rule condition ‘IF(RELATION hasToken ‘**bought**’)’.

Step 2: As the operator is ‘hasToken’, the user can click the ‘Check Synonym’ button then the top 10 synonyms from the WordNet automatically are displayed.

Step 3: The user can deselect inappropriate synonyms and retain relevant synonyms like ‘acquire’, ‘gain’.

Step 4: The user selects the rule conclusion ‘THEN (TRUE, VERB, buy)’.

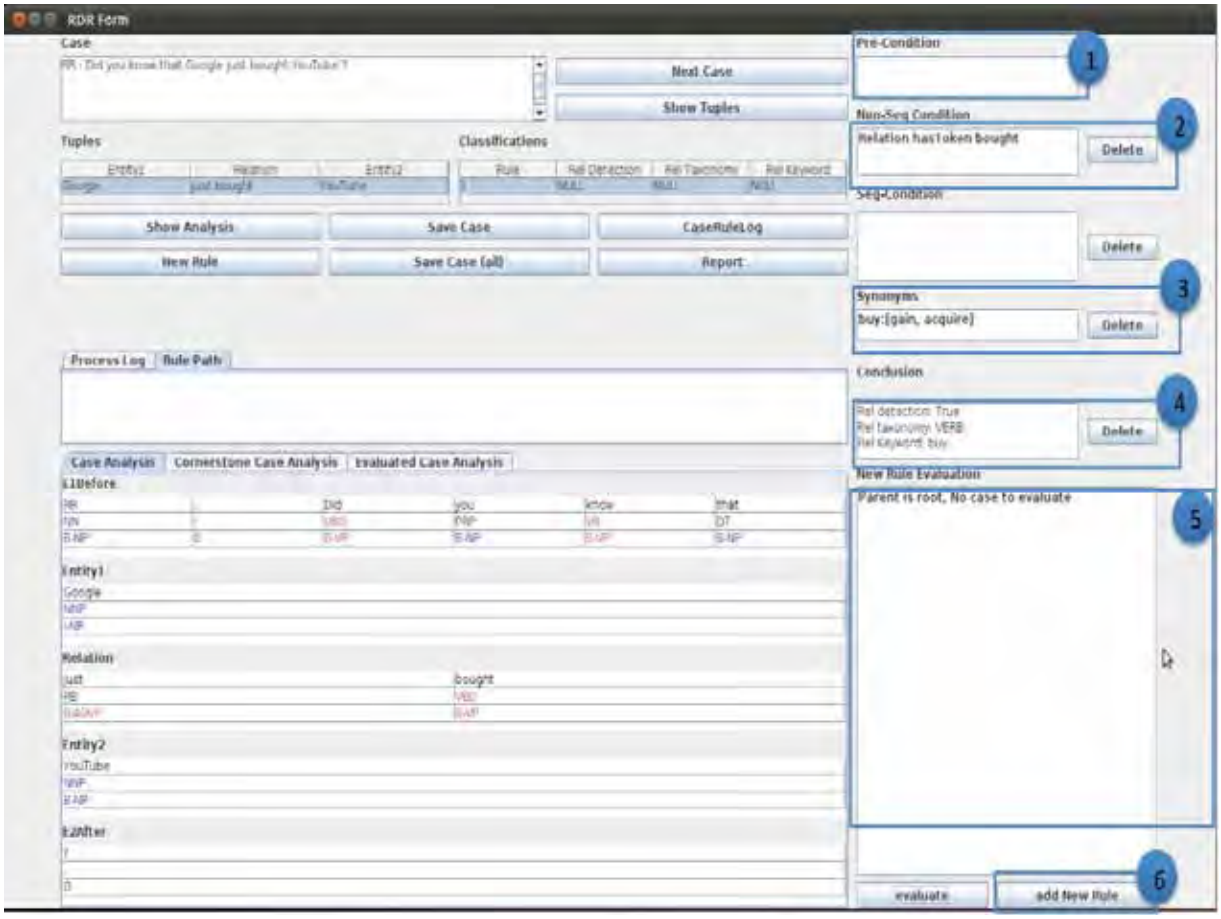

Figure 4.5. The KA process when evaluating the new rule created and saving it into the KB

Step 1: Pre-Condition is empty because the parent rule of the new rule R2 is the default rule R1.

Step 2: Non-sequence condition ‘IF(RELATION hasToken ‘**bought**’)’ is displayed.

Step 3: Chosen synonyms such as ‘gain, acquire’ are displayed.

Step 4: Conclusion ‘THEN (TRUE, VERB, buy)’ is displayed.

Step 5: Before saving the new rule into the KB, the user clicks the ‘evaluate’ button and the evaluation result ‘Parent is root. No case to evaluate.’ is displayed.

Step 6: As the new rule R2 has no conflict with the existing KB, the user adds the new rule R2 under the default rule R1.

After Case1 is processed successfully, Case 2 is entered into the system.

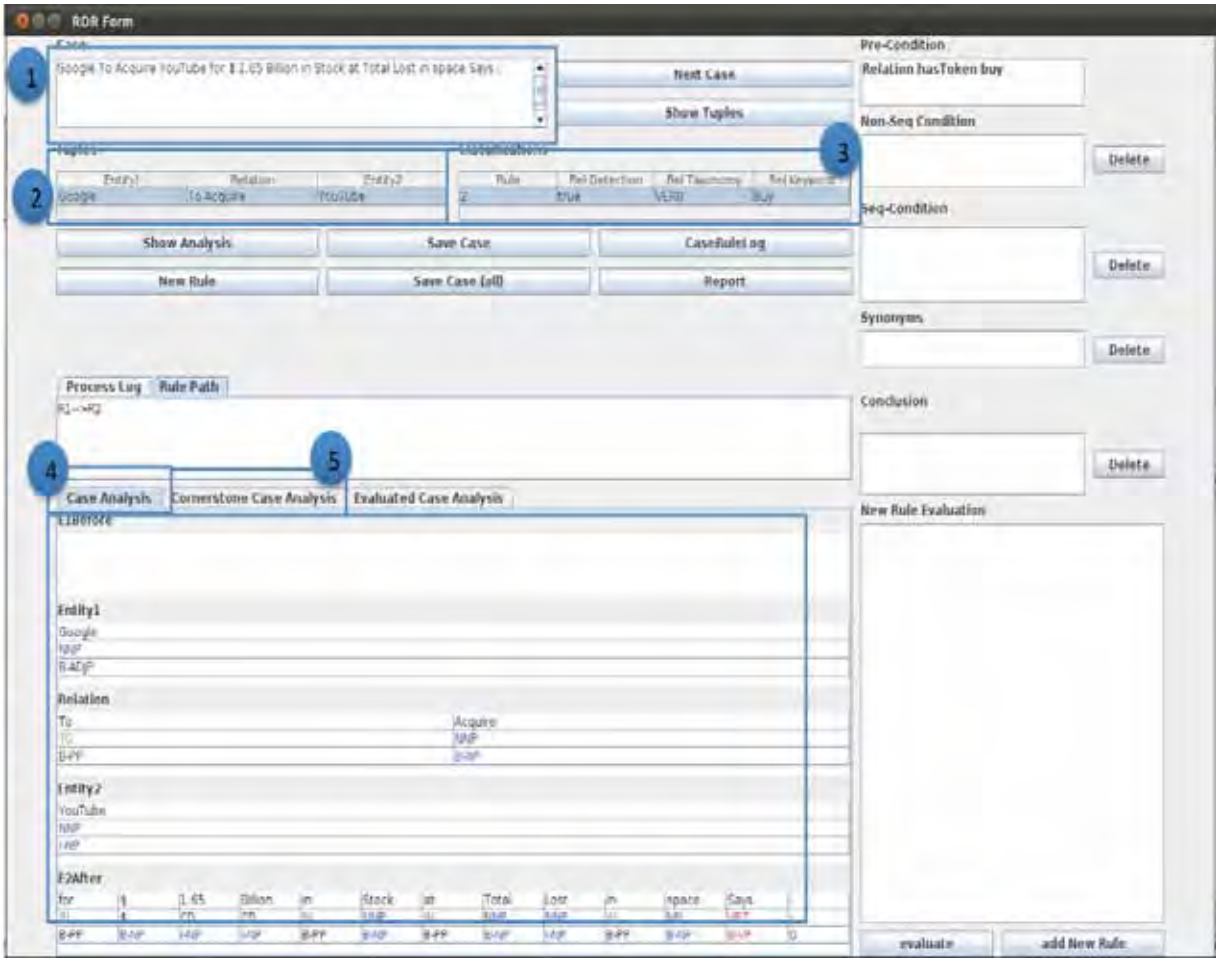

Figure 4.6. The KA process when a new case entered into the system

Step 1: The given sentence '*Google To Acquire YouTube $ 1.65 Billion Stock at Total cost in space Says:*' is displayed.

Step 2: The user clicks 'Show Tuples' button then a candidate tuple (**Google** , **To Acquire** , **YouTube**) extracted from the Tuple Extractor is displayed.

Step 3: As the user clicks the extracted tuple, the rule R2 fires with the '(True, VERB, buy)' conclusion from the KB. The user decides that the result is not correct as the given case's

taxonomy type should be INFINITIVE rather than VERB and decides to add an exception rule to correct it. This rule is automatically added under R2 as its child rule.

Step 4: The user clicks ‘Show Analysis’ button and the system presents NLP features such as token, part-of-speech and chunk phrase. The user checks the ‘Case Analysis’ table to figure out relevant features to make a new rule.

Step 5: The user checks the ‘Cornerstone Case Analysis’ table to differentiate features between the current case and the cornerstone case of the fired rule.

As the classification result is incorrect, the user clicks the ‘New Rule’ button to create an exception rule under R2, then the ‘RDR New Rule Form’ window pops up.

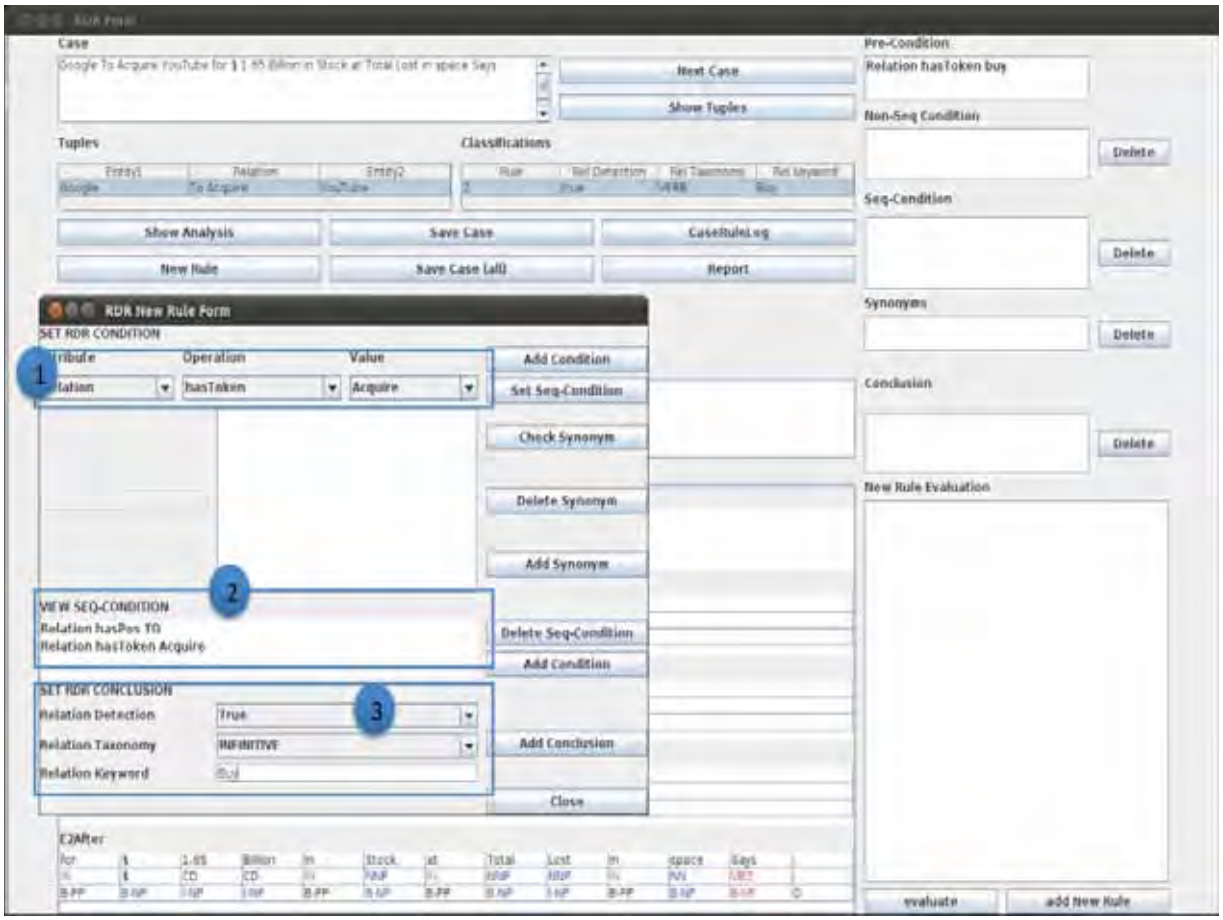

Figure 4.7. The KA process when the user selects a new rule’s condition and conclusion

Step 1: The user selects a rule condition.

Step 2: The user generates the sequence condition ‘IF SEQ((RELATION hasPos ‘**TO**’) & (RELATION hasToken ‘**Acquire**’))’ for the rule.

Step3: The user selects the rule conclusion as ‘THEN (True, INFINITIVE, buy)’.

After creating an exception rule, close the popped ‘RDR New Rule Form’ window.

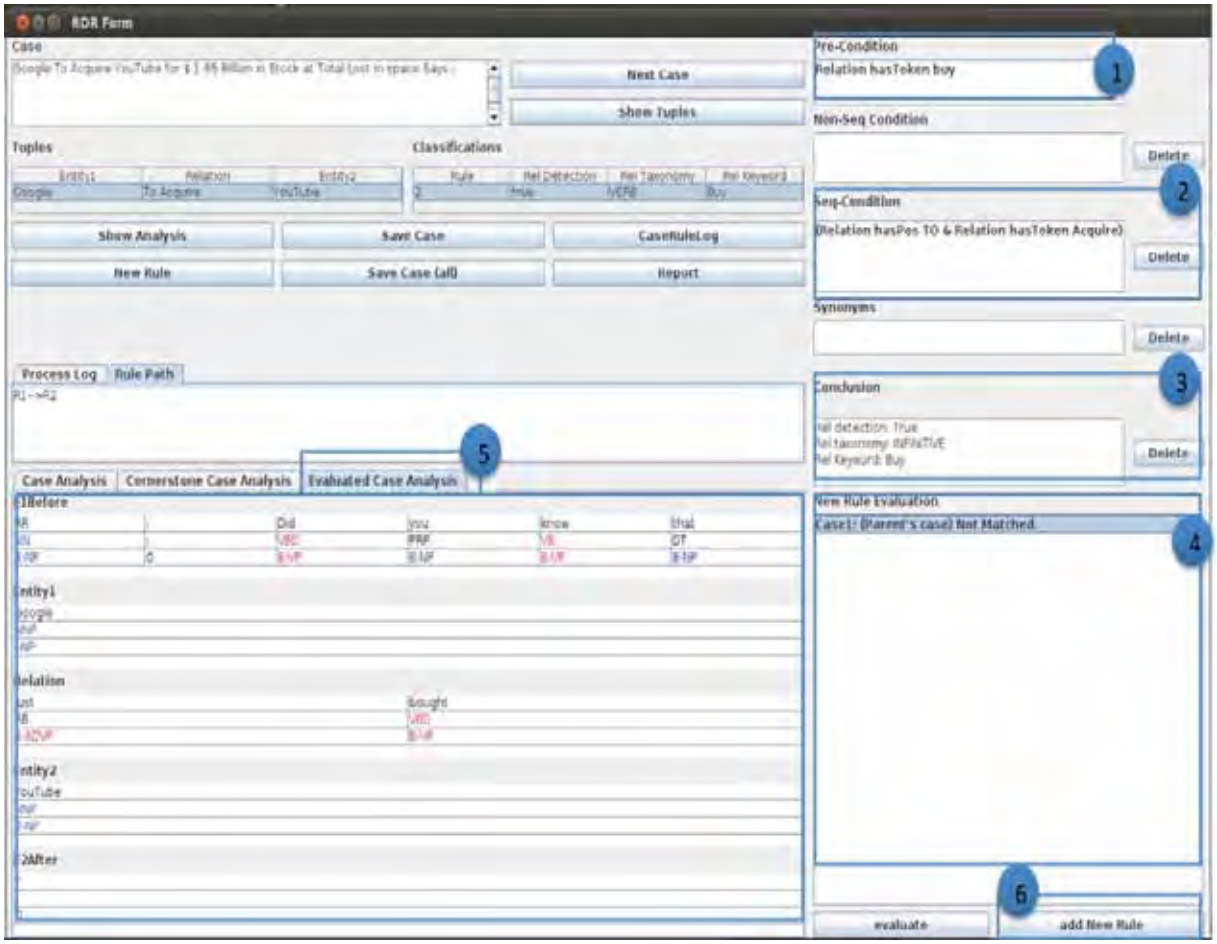

Figure 4.8. The KA process when evaluating the new rule and saving it into the KB

Step 1: The Pre-Condition section displays the condition of the parent rule R2 ‘(RELATION hasToken ‘**bought**’)’.

Step 2: The sequence condition ‘IF SEQ((RELATION hasPos ‘**TO**’) & (RELATION hasToken ‘**Acquire**’))’ is displayed.

Step 3: The conclusion ‘THEN (True, INFINITIVE, buy)’ is displayed.

Step 4: Before saving the new exception rule R3 into the KB, the user clicks the ‘evaluate’ button and the evaluation result ‘Case1: (Parent’s case) Not matched.’ is displayed.

Step 5: In case of any conflict occurs during evaluation, the user can check the ‘Evaluated Case Analysis’ table where each of evaluated case’s features is displayed. Through this table, the user can compare feature differences between the current case and the conflicted case, then refine the new rule not to conflict. Here, no conflict has been occurred.

Step 6: As the new exception rule has no conflict with the existing KB, the user adds the new exception rule R3 under the parent rule R2.

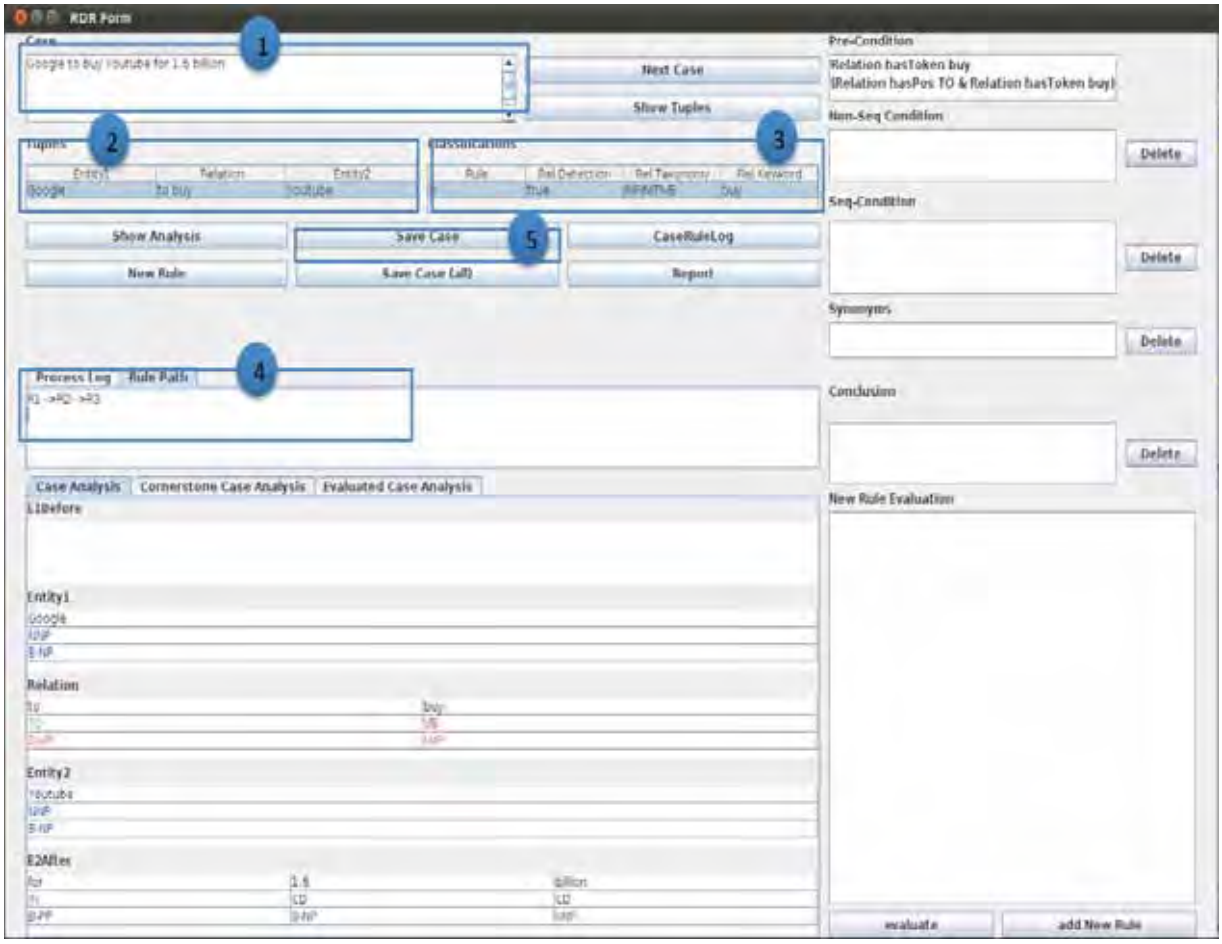

Figure 4.9. The KA process when a new case is entered in the system. After Case2 is processed successfully, Case 3 is entered into the system

Step 1: The given sentence '*Google to buy YouTube for 1.6 billion*' is displayed.

Step 2: The user clicks 'Show Tuples' button then a candidate tuple (**Google** , **to buy** , **YouTube**) extracted from the Tuple Extractor is displayed.

Step 3: As the user clicks the extracted candidate tuple, and rule R2 fires with '(True, INFINITIVE, buy)' conclusion from the KB. The user justifies the result as correct.

Step 4: The current rule path 'R1→R2→R3' is displayed.

Step 5: The user saves the current case under rule R3 by clicking ‘Save Case' button.

4.4 Experimental Setup

Experiments were conducted on two Web datasets referred to as ‘Sent500’ and ‘Sent300’. In previous work Sent500 [5] was used to evaluate the performance of the OIE systems TEXTRUNNER (Banko and Etzioni, 2008) and StatSnowball (Zhu et al., 2009). We use it as a test dataset to compare the performance of the RDROIE system to the published results from these other OIE systems. Sent500 is partially, and Sent300 is fully, derived from the MIL dataset[6] developed by Bunescu and Mooney (2007). Sent300 is a set of 300 sentences that we randomly selected from the 4155 argument pair sentences in the MIL dataset to be used as training examples to construct the initial KB. Bunescu and Mooney (2007) collected a bag of sentences by submitting a query string ‘a1 ******* a2’ containing seven wildcard symbols between the given pair of arguments mainly to find ‘corporate acquisition’ and ‘personal-birthplace’ relations. Each sentence selected has one pair of entities manually identified. The pairs of arguments used were ‘adobe systems and macromedia’, ‘google and youtube’, ‘novartis and eon labs’, ‘pfizer and rinat neuroscience’, ‘viacom and dreamworks’, ‘yahoo and inktomi’, ‘andre agassi and las vegas’, ‘chalie chaplin and london’, franz kafka and prague’, ‘george gershwin and new york’, ‘luc besson and paris’, and ‘marie antoinette and vienna’. Sent500 contains some randomly selected sentences from the MIL dataset and some more sentences for ‘inventors of product’ and ‘award winners’ relations, which were collected by the

[5] Available at ‘Available at http://www.cs.washington.edu/research/knowitall/hltnaacl08-data.txt’
[6] Available at ‘http://ace.cs.ohiou.edu/~razvan/’

TEXTRUNNER team using the same technique as used for MIL datasets. It should be noted that our training data set Sent300 came only from the MIL dataset and did not include any sentences in Sent500 test data.

4.5 Knowledge Base (KB) Construction

This section presents the analysis of the initial KB construction using the RDROIE system. In processing the Sent300 training set and adding rules as required, 43 new rules were created for the cases which received a NULL classification result and 9 exception rules were created for the cases which received incorrect classification result from earlier rules. In total, 52 rules were added within two and half hours. KB construction time covers from when a case is called up until a rule is accepted as complete and is logged automatically.

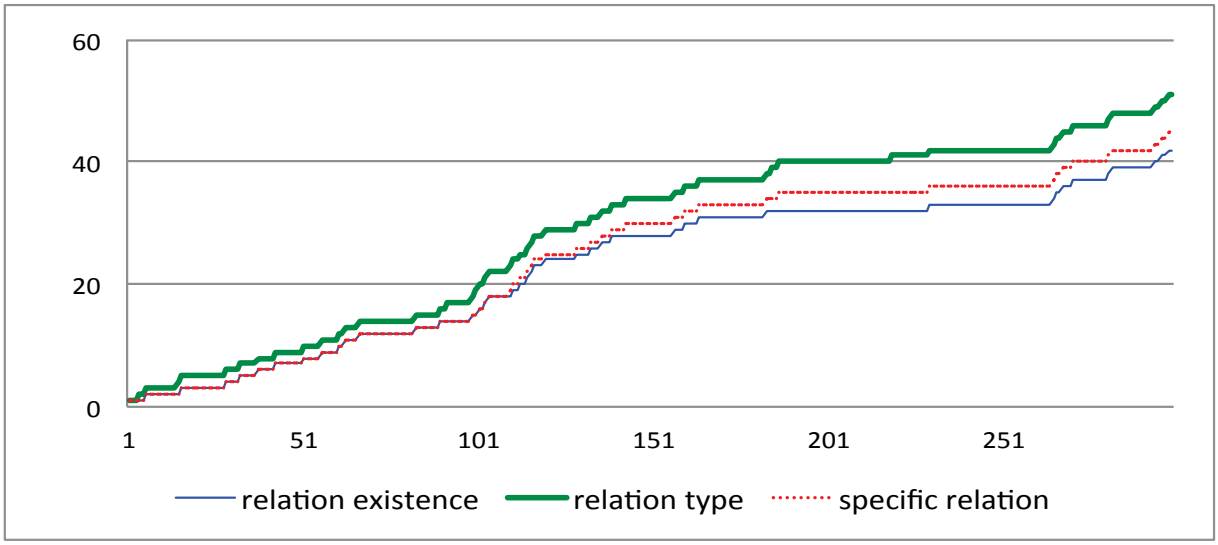

Figure 4.10. Total number of RDR rules added as plotted against the number of cases seen during training

In this experiment, the RDROIE system covers all three types of task: relation existence detection, relation taxonomy type detection and specific relation detection. Figure 4.10 shows the number of rule additions with respect to the number of cases processed for the three tasks. For the first 150 cases, 34 rules were created, but for the next 100 cases only 8 more rules were added, then for the last 50 cases 10 rules were added as shown in Figure 4. Although there is clear convergence after the first 150 cases until about 250, knowledge acquisition then accelerates again as novel cases are encountered in the system. This suggests that learning may be some way from complete after training on 300 sentences; however, this is the knowledge base used for comparison with the other methods.

From the analysis of the exception rules added, we categorise the causes of exceptions into two classes.

(1) An exception rule simply reduces the scope of the parent rule, making it more specific. The following is a simple example where the exception is that the parent rule should not apply if the verb is an infinitive. The parent rule applies to a verb, whereas the exception rule applies to particular types of verb, providing an increase in specificity:

Sentence	*Google to buy YouTube for $ 1.6bn - reports (6 October 2006)*
Tuple	(**Google**, **to buy**, **YouTube**)
Fired rule A	IF (RELATION hasToken '**buy**') THEN (True, Verb, 'buy')
Exception rule B	IF SEQ((RELATION hasPOS '**TO**') & (RELATION hasToken '**buy**'))

	THEN (True, Infinitive, 'buy')

(2) The exception here is more complex. As with all exception rules it reduces the scope of the parent rule but it does not provide a conclusion that is related to the parent. Rather the parent rule should not cover this type of case at all:

Sentence	*Charlie Chaplin , who died in 1977 , was born in London to music - hall parents .*
Tuple	(**Charlie Chaplin**, **, who died in 1977 , was born in**, **London**)
Fired rule A	IF SEQ((RELATION hasToken '**bear**') & (RELATION hasToken '**in**')) THEN (True, Verb+Prep, 'born in')
Fired rule B	IF SEQ((RELATION hasToken '**die**') & (RELATION hasToken '**in**')) THEN (True, Verb+Prep, 'die in')
Exception rule C (under the rule B)	IF SEQ((RELATION hasToken '**,**') & (RELATION hasToken '**who'**) & (RELATION hasToken '**die'**) & (RELATION hasToken '**in**') & (RELATION hasGap **1**) & (RELATION hasToken '**,**')) THEN (False, NULL, NULL)

In this case, two rules, rule A and rule B were fired; rule A extracted the ‘born in’ relation correctly but rule B extracted the ‘die in’ relation which is incorrect because the ‘die in’ relation is located in an dependent clause which has no semantic relation with the entity2 ‘London’. The exception rule C is added under rule B using the more specific condition.

4.6 Open Information Extraction (OIE)

4.6.1 OIE Systems Performance Comparison

We compared the performance of the RDROIE system with the state-of-the-art OIE systems, the TEXTRUNNER (O-CRF version) system and the StatSnowball system on the Sent500 data set.

The RDROIE system had been trained with Sent300, which contains 300 sentences randomly selected from the MIL dataset. TEXTRUNNER had been trained with 91,687 positive examples and 96,795 negative examples derived from the Wall Street Journal (WSJ) dataset. The StatSnowball system can perform both traditional IE and Open IE. The StatSnowball system accepts tuples with two entities as input such as (E1, E2, key relation) for traditional IE and (E1, E2, ?) for OIE to find relation patterns through bootstrapping. In the StatSnowball system, 30 sentences are randomly selected as initial seeds for bootstrapping and the results are then averaged over 10 runs. Each run could have various bootstrapping iterations e.g. from 4 to 10 (Zhu et al., 2009). In the StatSnowball system, two different pattern selection methods, the l1-norm regularised pattern selection and a heuristic-based pattern selection were used and in the table they are referred to as l1StateSnowball and heStatSnowball, respectively. We compared performance on four categories of relations: VERB, NOUN+PREP, VERB+PREP and INFINITIVE as evaluated in (Banko and Etzioni, 2008; Zhu et al., 2009). The experiment results

are shown in Table 4.1, where the results of TEXTRUNNER and StatSnowball systems are from the original paper comparing the two systems (Zhu et al., 2009).

Category		VERB	NOUN+PREP	VERB+PREP	INFINITIVE	All
RDROIE	P	**0.98**	**0.96**	**0.99**	**1.00**	**0.98**
	R	0.56	0.47	**0.82**	**0.93**	0.63
	F1	0.71	0.63	**0.90**	**0.96**	**0.77**
TEXTRUNNER	P	0.94	0.89	0.95	0.96	0.88
	R	0.65	0.36	0.50	0.47	0.45
	F1	0.77	0.51	0.66	0.63	0.60
heStateSnowball	P	0.96	0.70	0.58	0.46	0.79
	R	0.81	0.50	0.34	0.19	0.58
	F1	0.88	0.54	0.40	0.23	0.67
l1StatSnowball	P	0.93	0.82	0.62	0.42	0.80
	R	**0.85**	**0.78**	0.58	0.32	**0.73**
	F1	**0.89**	**0.80**	0.60	0.36	0.76

Table 4.1. Evaluation result of different OIE systems on Sent500

(P= Precision, R=Recall, F1=F-measure)

Overall, the RDROIE system outperforms the TEXTRUNNER system and the heStatSnowball system for all four categories for precision, recall and F1. The RDROIE system also achieves much higher precision over the four individual categories. For “VERB+PREP” and

"INFINITIVE", the RDROIE system performs better than the other OIE systems for precision, recall and F1. On the other hand, the RDROIE system shows the lowest recall for the "VERB" category and lower recall for "NOUN+PREP" category compared to both versions of the StatSnowball system.

	Incorrect classification	NULL classification
Verb	2	77
Noun+Prep	2	55
Verb+Prep	1	16
Infinitive	0	3
All	5	151

Table 4.2. Error cases on four categories of the RDROIE system

Table 4.2 shows an error case analysis for the RDROIE system over the four categories. 5 cases were incorrectly classified under another class and 151 cases were incorrectly classified as NULL. Most of cases with NULL classification should have been classified under the "VERB" and "NOUN+PREP" categories which explains the low recall of the RDROIE system for these categories. Obviously the training data did not contain relevant examples, but since the approach allows a user to fix error cases easily while the system is already in use, the low recall could be improved quickly in real world usage.

In a further analysis we divided the knowledge base into three groups of rules: the first 20 rules, the first 40 rules and finally all 52 rules, and evaluated precision, recall and F1 score on

Sent500. Figure 4.11 shows the performance of the RDROIE system with respect to the number of rules as the KB size increases. From the results, we can see precision is very high even after 20 rules, and stays high as more rules are added. Further rules do not degrade precision but improve recall and F1 score, suggesting an RDR approach does not trade precision for recall.

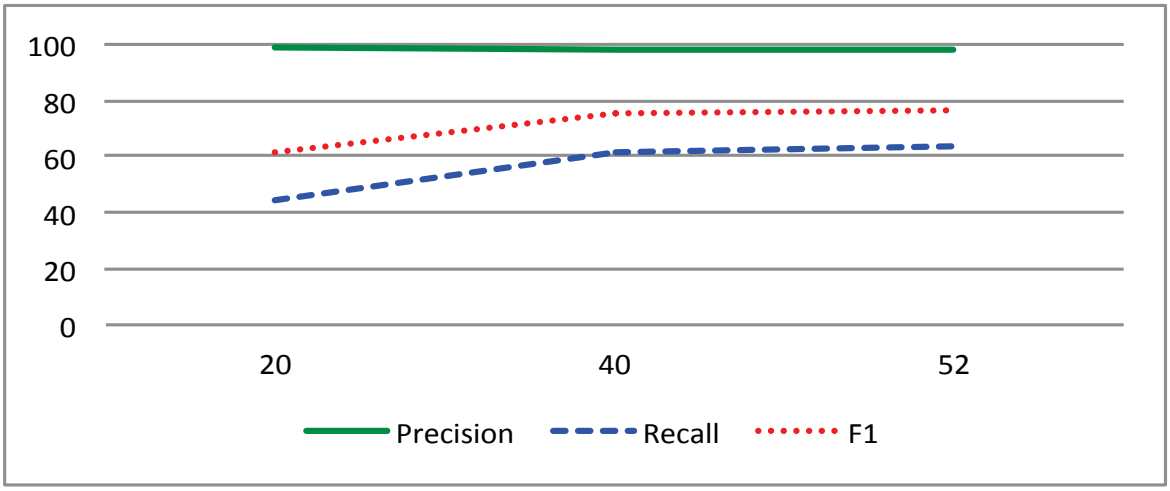

Figure 4.11. The performance of the RDROIE system with incrementally added rules

4.6.2 Advantages of the RDROIE System

There are number of key features of the RDROIE system: (1) handling the free writing style that occurs in Web documents (2) handling NLP tools errors (3) handling abstract expressions (4) handling negative expressions.

4.6.2.1 Handling the Free Writing Style of Web Documents

Most Web documents are written without strict supervision. For example, there are many spelling errors on the Web.

The following is an example of a spelling error and an RDR rule to handle it.

Cases	'*Mind Booster Noori : Google aquires YouTube*' '*__ is self sufficient , and Google aquiring YouTube is a sure sign of this*'
Issue	'acquire' misspelled as 'aquire'
Rule	IF (RELATION hasToken '**aquire**') THEN (True, Verb, acquire)
Cases	'*George Gershwin ' s " Porgy and Bess " premiers at Alvin Theater , New York City : 1935 [African & Black History]*'
Issue	'premiere' misspelled as 'premier'
Rule	IF (RELATION hasToken '**premiers**') THEN (True, Verb, premiere)

The following is an example of a Web-specific abbreviation and an RDR rule to cover it:

Cases	'*(Encyclopedia) Kafka , Franz , 1883 & ndash1924 , German - language novelist , b. Prague* .'
Issue	'b.' is used to indicate 'born'
Rule	IF (RELATION hasToken '**b.**') THEN (True, Verb, born)

4.6.2.2 Handling NLP Tools Errors

As most NLP tools are trained on journalistic text, which follows strict writing rules; they do not perform as well on Web documents written in a freer style. For example the informal usage of capitalisation often confuses NLP tools.

The following is an example of NLP tool error and an RDR rule to handle it:

Cases	'*Yahoo Rejects Microsoft Search Offer*' '*Google Nabs YouTube For $ 1.65bn*' '*GOOGLE BUYS YOUTUBE*'
Issue	'Rejects', 'Nabs' and 'BUYS' tagged incorrectly as proper noun rather than verb
Rule	IF (RELATION hasToken '**BUYS**') THEN (True, Verb, buy)

Part-of-speech tagging errors often cause degradation of OIE system performance (Banko and Etzioni, 2008). Most OIE systems are reliant on POS tags and for example, cannot correctly identify a relationship when a verb is incorrectly tagged as a noun.

4.6.2.3 Handling Abstract Expressions

There are some abstract expressions that require translation or understanding of the vernacular.

The following is an example of an abstract expression and an RDR rule to handle it:

Cases	'*Google just ate YouTube .*' '*Google swallow YouTube .*'

Tuples	(**Google** , **just ate** , **YouTube**) (**Google** , **swallow** , **YouTube**)
Issue	It is better to save the ‘acquire’ relation rather than ‘ate’, ‘swallow’ relation in KB.
Rule	IF (RELATION hasToken ‘**ate**’) & (ENTITY1 hasToken ‘**Google**’) & (ENTITY2 hasToken ‘**YouTube**’) THEN (True, Verb, ‘acquire’) OR IF (RELATION hasToken ‘**ate**’) & (ENTITY1 hasToken ‘**Google**’) & (ENTITY1 hasNE ‘**ORG**’) THEN (True, Verb, ‘acquire’) [‘swallow’ is automatically recommended as a synonym of ‘eat’ in the user interface]

4.6.2.4 Handling Negation Expressions

None of OIE systems yet handle negative expressions.

The following is an example of a negative expression and a RDR rule to cover it:

Case	‘*In that ruling in December 2003 , Judge Richard G. Stearns dismissed Tevas lawsuit because Pfizer hadn’t accused Teva of infringing its Zoloft patent and __ had no reasonable basis to expect __ would do*

	so .'
Tuple	(**Pfizer** , **had not accused** , **Teva**)
Issue	Extract negation relation.
Rule	IF (RELATION hasToken '**hadn't accused**') THEN (True, Verb, not accused)

4.7 Discussion

Figure 4.11, shows that RDROIE system rapidly achieves and maintains a very high level precision after only 2.5 hours training by a user on a small dataset and overall outperforms systems using sophisticated automated training algorithms. The question immediately arises whether more appropriate training data was used with the RDROIE system. There was no overlap between the test and training set, but obviously we used relevant training data. However, this illustrates the advantage of an RDR approach. The RDROIE system can be trained on whatever text it is meant to classify and errors can keep on being corrected at minimal cost while it is in use. In contrast machine learning-based methods require very large amounts of training data, and there may be no suitable training data closely similar to the domain where the system is to be used.

The RDROIE system requires very little effort; rule creation is very simple and rapid. In the study here it took about three minutes on average to build a rule. Experience suggests that knowledge acquisition with RDR remains very simple and rapid even for large knowledge bases. Rules can be updated as errors are uncovered, or when new vocabularies are created, or new

meanings are attached to existing terminologies, or new colloquialisms come into use. The alternative is to accumulate enough training data to retrain a system automatically. An error can be corrected manually the first time it occurs, but with a machine learning system, one needs to have many examples.

There is an important underlying assumption in the RDROIE system: that it is more useful to have a system that works well on any specific domain of interest, and able to deal with all the idiosyncrasies of language and expression in the domain, when they are observed, rather than trying to have a general system for all domains, but which cannot deal with the NLP errors and other oddities that may arise in a specific domain of interest. Ideally, one would like a system that can deal with anything, but TEXTRUNNER has already been trained on 200,000 labelled sentences. How many sentences from what corpora would be needed to have a system for any and all occasions? The RDROIE system is certainly not a system for all occasions, but we suggest it is a system for any occasion. That is, rather than using a general solution, one can very rapidly build a new system to deal with the specific problems of the domain of interest. In the experiment, the RDROIE system used only 300 Web sentences as training examples and clearly the training set did not include some of relations that appeared in the test set. The point of the RDROIE approach is that when these omissions occur they can be corrected.

There is likely to be a value in combining RDR and machine learning approaches (Xu and Hoffmann, 2010; Ho et al., 2009), where rather than building the entire system using RDR, machine learning is used to build as comprehensive approach as possible but then errors and omissions are handled using RDR. We will discuss a version of this idea in the coming chapters where RDR is used on top of an existing system; however, we wish to emphasise that in the

study here, a human was able to outperform machine learning systems after 2.5 hours knowledge acquisition.

Chapter 5. Hybrid Ripple-Down Rules based Open Information Extraction (Hybrid RDROIE)

Chapter 4 demonstrated how we can build an RDR-based OIE system that outperformed previous machine learning-based OIE system, TEXTRUNNER on Web data in a narrow range of interest. Although the RDROIE system has not been tested on data outside the range of interest necessarily it will perform worse than something like TEXTRUNNER, the general OIE system.

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 the general system performance. The aim of this chapter is to test this hypothesis.

In chapter 4, we conducted the experiments on Sent500 data to be able to make a comparison with published data on the TEXTRUNNER system as no OIE system was publicly available at the time these experiments were conducted. Recently, the REVERB system (Fader et al., 2011) has been introduced and has become publicly available. As the recent REVERB system is the only publicly available OIE system, we determined to adopt it as a general OIE system for our Hybrid RDROIE system.

Fader et al. (2011) presented the problems of state-of-the-art OIE systems such as the TEXTRUNNER system (Banko and Etzioni, 2008) and the WOE system (Wu and Weld, 2010) 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 is a 'relation first' rather than an 'arguments first' system, to try to avoid the common errors of previous OIE systems. REVERB achieved an AUC[7] that is 30% higher than WOE*parse* and is more than double the AUC of WOE*pos* or TEXTRUNNER (Fader et al., 2011). The REVERB system outperformed the others on the test dataset, which contained 500 sentences sampled from the Web, using Yahoo's random link service. [8] The Yahoo random links does not accept all sites to be linked, therefore it is more likely to find a quality website since they are all screened (http://www.mangle.ca/ranlinks.php).

Although the REVERB system showed good performance on their Web dataset, we found that its performance significantly dropped on the Sent500. This is because their Web dataset does not contains much informality since it is sampled from the formal Web sites accepted by the Yahoo

[7] Area Under the Curve computed by a precision-recall curve by varying confidence threshold
[8] http://random.yahoo.com/bin/ryl

random link. On the other hand, the Sent500 was collected from Google search engine by submitting a query string ‘a1 ****** a2’ containing seven wildcard symbols between the given pair of arguments, thus it contains more informally written sentences, spelling errors, various symbols and noise. Section 5.1 presents the performance of the REVERB system on the Sent500.

In this chapter, we describe the Hybrid RDROIE system where the REVERB system is used as a base OIE system and the Ripple-Down Rules (RDR) technique is applied to deal with the Web’s informality for the OIE task. The underling idea of the Hybrid RDROIE system is that it takes advantage of the state-of-the-art performance from the REVERB system but then corrects any errors incrementally while the system is in routine use. In the Hybrid RDROIE system, each relation tuple extracted from the REVERB system is run through KB. When the default rule R1 is fired with NULL classification and the given case (a tuple) from the REVERB system is correct, then the system simply saves the case as correct relation extraction. On the other hand, when the given case (a tuple) from the REVERB system is incorrect or, when the classification result from KB is incorrect, the user creates rules to correct it. As the rules are generated by humans, the Hybrid RDROIE system is more likely to be able to handle OIE errors due to informal Web documents. After two hours training in a small training dataset, the Hybrid RDROIE system achieved almost double the performance of the REVERB system.

Section 5.1 demonstrates how the Web’s informality challenges existing the state-of-the-art REVERB OIE system and section 5.2 describes the Hybrid RDROIE system architecture. Section 5.3 presents a description of RDR rules and examples of the MCRDR structure for both NLPRDR KB (RDR KB to correct NLP errors) and TupleRDR KB (RDR KB to correct tuple errors). Section 5.4 demonstrates the knowledge acquisition process of the Hybrid RDROIE system through user interface screenshots. Section 5.5 presents the experimental setup and

section 5.6 describes the two initial knowledge bases (NLPRDR KB and TupleRDR KB) constructed with the Hybrid RDROIE system. Section 5.7 demonstrates how the Hybrid RDROIE handles the errors of the REVERB system on the Web dataset. Section 5.8 presents the improved performance of the Hybrid RDROIE system from the base system, the REVERB system. Lastly, section 5.9 discusses these results.

5.1 REVERB Error Analysis on 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', which was used to evaluate the performance of OIE systems in chapter 4. Originally, in this dataset, each sentence has one pair of entities manually identified for the relation extraction task, but those entity tags are removed in this experiment. That is, there are no pre-defined tags in the Sent500 dataset used here.

Extractions are judged by the following conditions. 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 the 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 the extraction. For example, the tuple extraction (**Another example of a statutory merger** , **is** , **software maker Adobe Systems acquisition of Macromedia**) is treated as an incorrect extraction while the tuple extraction (**Adobe** , **has announced the acquisition of** ,

Macromedia) is counted as a correct extraction. N-ary relations such as (**Google** , **has officially acquired YouTube for** , **$ 1.65 bil**) are also treated as a correct extraction.

	All	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 5.1. REVERB performance on Sent500

Table 5.1 shows the performance of REVERB overall and on four different classes. The overall result is evaluated based on all extractions from Sent500 using REVERB, while the four categories results are evaluated based on extractions of the pre-tagged entities of Sent500 and its relations. The result on all extractions shows that overall the REVEB performance on Sent500 is quite poor at around 40%. In the four categories result, VERB and VERB+PREP categories show higher precision than the NOUN+PREP and INFINITIVE categories. Especially, the recall of NOUN+PREP and INFINITIVE categories are very low, 26.13% and 20.45%, respectively. This is because that the REVERB system aims to extract binary relations expressed by verbs.

	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 5.2. Incorrect extraction errors analysis on each category

Table 5.2 summaries the types of incorrect extraction errors for four categories. In VERB and VERB+PREP categories, most of the false positive errors, 84% and 91% respectively, are due to incorrect entity detection while relation detection is correct. As the REVERB system extracts entities using noun phrases, which are located nearest to the detected relation, it often recognises 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 the 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 NOUN+PREP and INFINITIVE categories, most of false positive errors, 55% and 67% respectively, are due to incorrect detection of both relations and entities.

	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 5.3. Missed extraction errors analysis on each category

Table 5.3 presents the types of missed extraction errors on four categories. In the VERB category, 72% of errors are due to NLP errors. For example, in a sentence '*Google Buys YouTube.*', the REVERB system missed an extraction because 'Buys' is tagged as a noun instead of a verb.

Due to the Web's informality, for example informally used capital letters, NLP tools often incorrectly annotate Web text. In the VERB+PREP category, 81% of the errors are due to unusual expressions. The REVERB system includes approximately 1.7 million distinct normalised relation phrases, which are derived from 500 million Web sentences. As the REVERB system 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 NOUN+PREP and INFINITIVE categories errors are mostly due to non-verb-based relation extraction. As the REVERB system aims to extract binary relations expressed by verbs, it only can extract NOUN+PREP and INFINITIVE type relations when there is a verb before a NOUN+PREP or INFINITIVE relation phrase. That is, when there exists tuples like (Entity1, verb NOUN+PREP, Entity2) or (Entity1, verb TO VB, Entity2), the REVERB system can extract NOUN+PREP or 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 the sentence '*Adobe About to Buy Macromedia .*' because there is no verb between the entities 'Adobe' and 'Macromedia'.

As discussed earlier, the REVERB system achieved more than double the AUC of TEXTRUNNER, but on Sent500, REVERB's performance shown is worse than TEXTRUNNER. Why is it? REVERB's main improvement is the elimination of incoherent extractions[9], because incoherent extractions were a large fraction of the errors made by previous systems such as TEXTRUNNER and WOE (Fader et al., 2011). We guess that Sent500 contains fewer sentences

[9] Incoherent extractions are cases where the extracted relation phrase has no meaningful interpretation (Fader et al., 2011).

which derive incoherent extractions compared to REVERB's test set. Also, we guess that TEXTRUNNER is more flexible then REVERB on informal text as TEXTRUNNER is based on machine learning-based method while REVERB is based on a relation phrase dictionary and constraint rules, which mainly reduce incoherent extractions. Additionally, REVERB aims to mainly extract verb-based relation while TEXTRUNNER extracts all types of binary relations.

The REVERB system showed very poor recall on Sent500. 89% and 95% of false negative errors (which affects recall) on VERB and VERB+PERP due to the Web's informality (NLP errors, noise and unusual expressions), respectively. Also, 93% and 100% of false negative errors on NOUN+PERP and INFINITIVE are due to non-verb relations, respectively. The Hybrid RDROIE system, is intended to handle these types of errors to improve the REVERB system.

5.2 Hybrid RDROIE System Architecture

The Hybrid RDR-based Open Information Extraction (Hybrid RDROIE) system shown in Figure 5.1 consists of four main components: preprocessor, NLPRDR KB learner, REVERB system and TupleRDR KB learner. The Hybrid RDROIE system initially has a preprocessor and an NLPRDR KB and then the TupleRDR KB. We considered that it is more efficient to clean up NLP errors before using the REVERB system for tuple extraction rather than just fixing errors after the REVERB system. This is because as shown above, REVERB's recall is very poor and one of main reasons is NLP errors. If we use the REVERB system first before applying NLPRDR KB, then we cannot improve the REVERB system's recall.

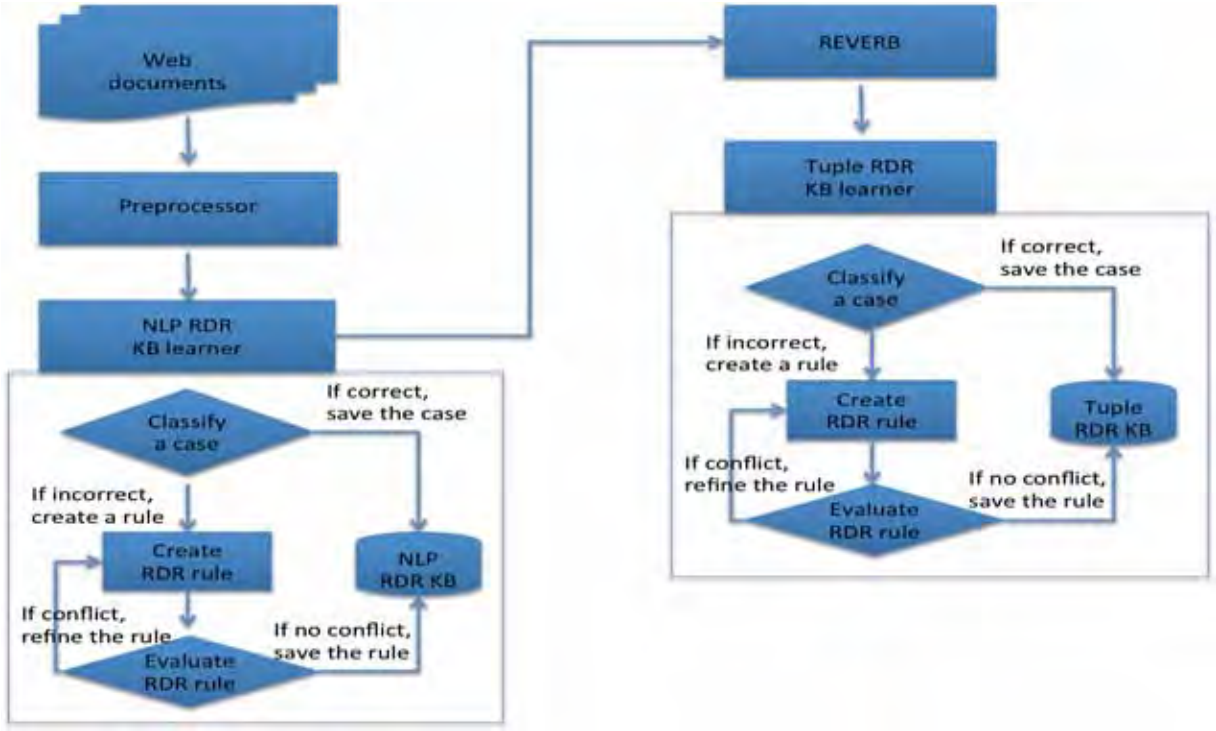

Figure 5.1. Architecture of the Hybrid RDROIE System

5.2.1 Preprocessor

The preprocessor converts the 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 for each token using the Stanford NER system. Annotated NLP features are utilised when creating RDR rules.

5.2.2 NLPRDR KB Learner

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

Step1: NLP classification

The NLPRDR KB takes each preprocessed sentence from the preprocessor and returns classification results. If RDR rules are fired and the fired rules deliver correct 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 it 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 new conclusion should apply to the previous case. Experience from other domains indicates that even if a large number of case fires the rule, the user will only have to consider two or three previous cases to select enough features to construct a sufficiently precise rule.

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

the REVERB system.

5.2.3 TupleRDR KB Learner

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

The TupleRDR KB is built incrementally while the system is in use with the REVERB system. 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 when the tuple classification result is incorrect, the user adds exception rules to correct it. The following three steps are used to construct the TupleRDR KB.

Step1: Tuple classification

The TupleRDR KB takes each tuple extraction from the REVERB system and returns classification results. If RDR rules are fired 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 actions. If the root rule is fired and the tuple is correct, then the only action is to save the correct extraction in the database.

Step2: Create RDR rule

Whenever incorrect classification 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 the RDR rule

Same as above (step3 in NLPRDR KB)

5.3 RDR Rule Description

An RDR rule comprises a condition part and a conclusion part. It has the form:

if CONDITION

then CONCLUSION

where CONDITION may include more than one condition.

5.3.1 Condition

A CONDITION consists of three components: ATTRIBUTE, OPERATOR and VALUE.

5.3.1.1 Attribute

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.

5.3.1.2 Operator

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 noun/verb chunk matches

• hasNE: whether a certain named entity matches

• hasGap: skip a certain number of tokens or spaces to generate a pattern

• notHasPOS: whether a certain part-of-speech does not match

• notHasNE: whether a certain named entity does not match

• beforeWD(+a): checks tokens positioned before the given attribute token by +a

• afterWD(+a): checks tokens positioned after the given attribute token by +a

5.3.1.3 Value

VALUE is derived automatically from the given sentence corresponding to the ATTRIBUTE and OPERATOR chosen by the user in the user interface. For example, in the NLPRDR KB, with the given sentence '*Google actually bought YouTube*', if the ATTRIBUTE is set to SENTENCE and the OPERATOR is set to 'hasChunk' then the VALUE 'B-NP, B-ADVP, B-VP, B-NP' will be displayed automatically in the user interface. In the TupleRDR KB, with the given tuple '(**Google** , **actually bought** , **YouTube**)', if the ATTRIBUTE is set to RELATION and the OPERATOR is set to 'hasPOS' then the VALUE 'NNP, RB, VBD, NNP[10]' will be displayed automatically in the user interface.

5.3.1.4 Type of Conditions

In both NLPRDR KB and TupleRDR KB, conditions are connected with an 'and' operation. A sequence of conditions begins with '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:

[10] See Appendix C for part-of-speech tag

‘SEQ((RELATION hasToken ‘**born**’) & (RELATION hasToken ‘**in**’))’

detects ‘born in’ in the RELATION element of the tuple.

5.3.2 Conclusion

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 such as SENTENCE, ENTITY1, RELATLION and ENTITY2

fixFrom, --- classification result before refinement

fixTo) --- classification result after refinement

5.3.3 Examples of Hybrid RDROIE Rules

The Hybrid RDROIE system is based on Multiple Classification RDR (MCRDR) (Kang et al.,

1995). Figure 5.2 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).

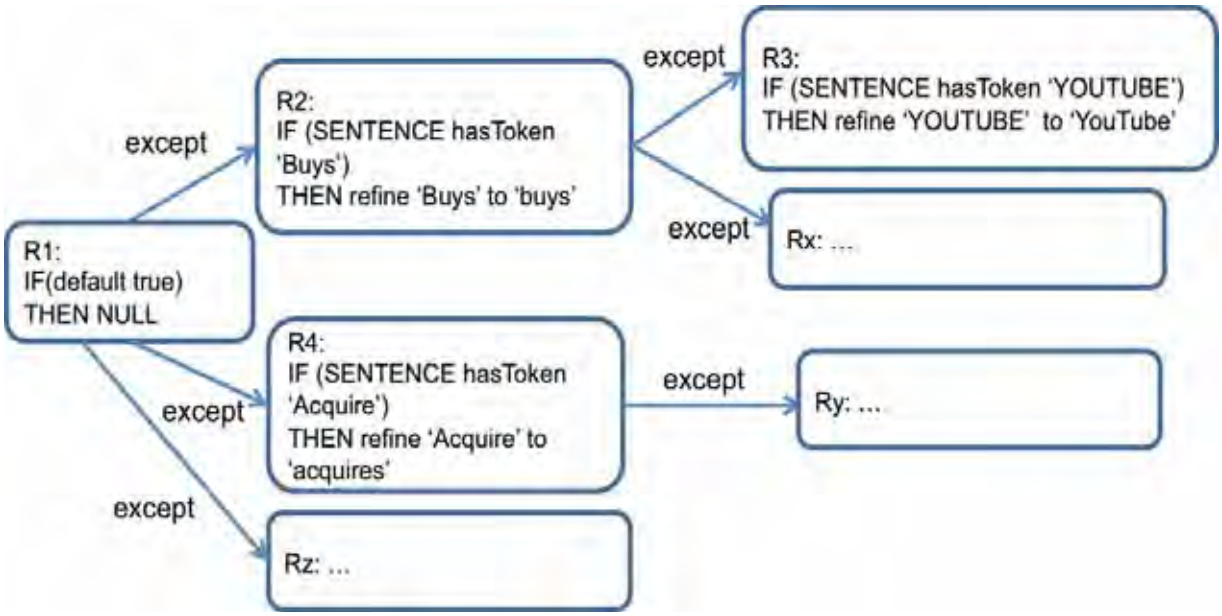

Figure 5.2. MCRDR Structure of the NLPRDR KB System

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

→ The default rule R1 is fired and the KB returns a NULL classification and the user considers it as 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 justifies the result as an incorrect classification result because 'YOUTUBE' should be refined to 'YouTube' to be correctly tagged by the NLP tools and extracted by the REVERB system.

→ A user adds an exception rule R3 under the parent rule R2.

Case3: A sentence '*Adobe system Acquire Macromedia.*'

→ The default rule R1 is fired and the KB returns a NULL classification, which the user considers as an incorrect classification result because 'Acquire' should be refined as 'acquires' to be extracted correctly by the REVERB system.

→ A user adds a new rule R4 under the default rule R1.

Figure 5.3 shows the MCRDR based KB construction as the TupleRDR KB system processes the following three cases (tuples) starting with en empty KB (with a default rule R1 which is always true and returns NULL classification).

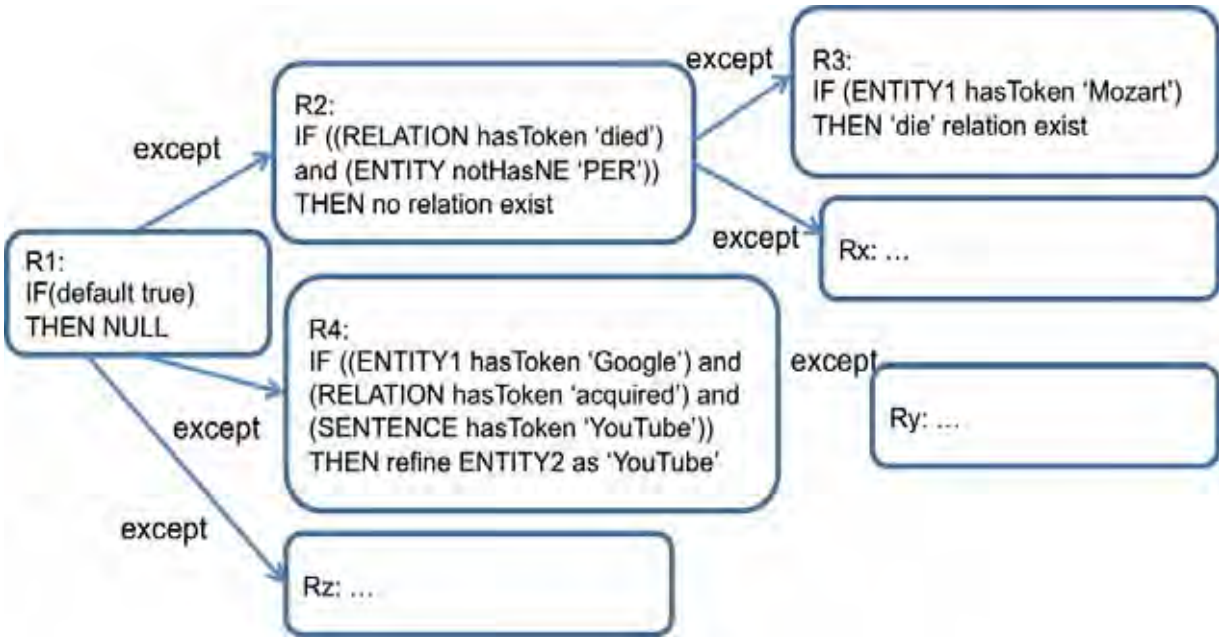

Figure 5.3. MCRDR Structure of the TupleRDR KB System

Case1: A 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 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: A tuple (**Wolfgang Amadeus Mozart** , **died** , **5 December 1791**) from the given sentence '*Wolfgang Amadeus Mozart died 5 December 1791.*'

→ Rule R2 is fired and classifies the given tuple as 'no relation tuple' and the user considers it as an incorrect classification result because the tuple contains a correct relation. This happens since the named entity tagger has not tagged the token 'Mozart' as a PERSON NE.

→ The user adds an exception rule R3 under the parent rule R2.

Case3: A 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 is fired and the KB returns a NULL classification, which the user justifies as an incorrect classification result because ENTITY2 contains 'the Video' instead of 'YouTube'.

→ A user adds new rule R4 under the default rule R1.

5.4 Knowledge Acquisition (KA) Process for the Hybrid RDROIE System

In this section, we present a number of screenshots of the Hybrid RDROIE system user interface to demonstrate the Knowledge Acquisition (KA) process starting with an empty KB. In the Hybrid RDROIE system, the user carries out the KA process through a friendly user interface, which automatically displays features needed in each KA process. On the screenshot, each KA process is numbered to indicate its order and each process step description is presented below each screenshot.

The Hybrid RDROIE system is written in Java (Java 1.6) and it used the OpenNLP system (version 1.5), the Stanford NER system (version 1.5) and the REVERB OIE system (version 1.1).

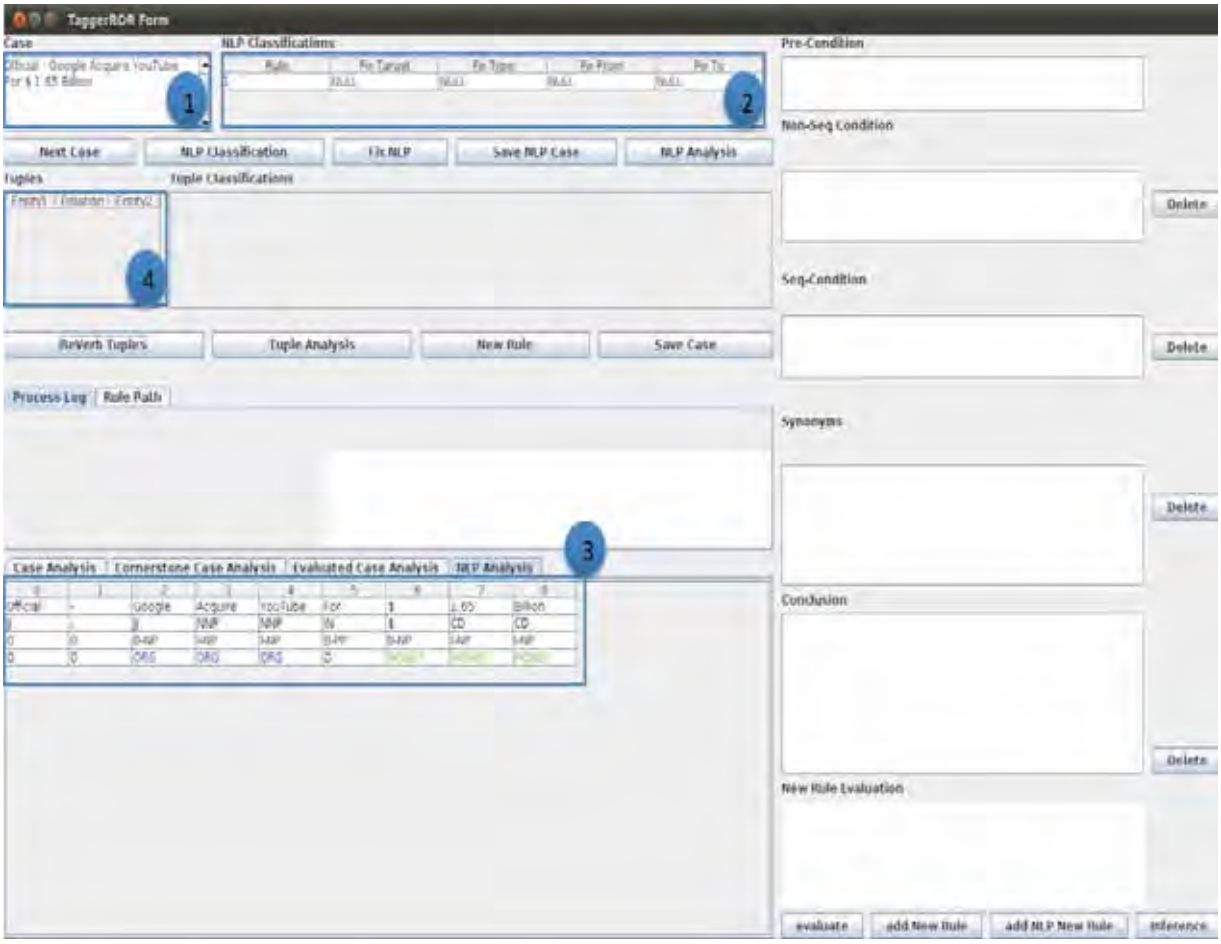

Figure 5.4. The KA process when a case (a sentence) is entered and the no relation tuple extraction is produced by the REVERB system due to NLP errors in the given sentence

Step1: The given sentence '*Official – Google Acquire YouTube for $ 1.65 Billion*' is displayed.

Step2: The user clicks the 'NLP Classification' button and the system automatically returns the classification result (the default rule R1 is fired and returns the NULL classification) from the NLPRDR KB (which is an empty at this stage).

Step3: Since the NLPRDR KB returns the NULL classification result, the user decides to check the features of the given sentence. The user clicks the 'NLP Analysis' button then the system displays NLP features such as part-of-speech tag, noun/verb chunk tag and named entity tag for

each token of the sentence. The user notices that the token ‘Acquire’ used a capital letter informally, which affects the relation extraction (tuple) of the REVERB system.

Step4: The user clicks the ‘REVERB tuples’ button. The REVERB system returns no tuples although the sentence includes binary relations to be extracted. The user decides to refine the case (sentence) to avoid the NLP error, in order that the REVERB system can correctly extract the binary relation tuple from the given sentence.

After the user clicks the ‘Fix NLP’ button on the main user interface, a small window pops up for rule creation.

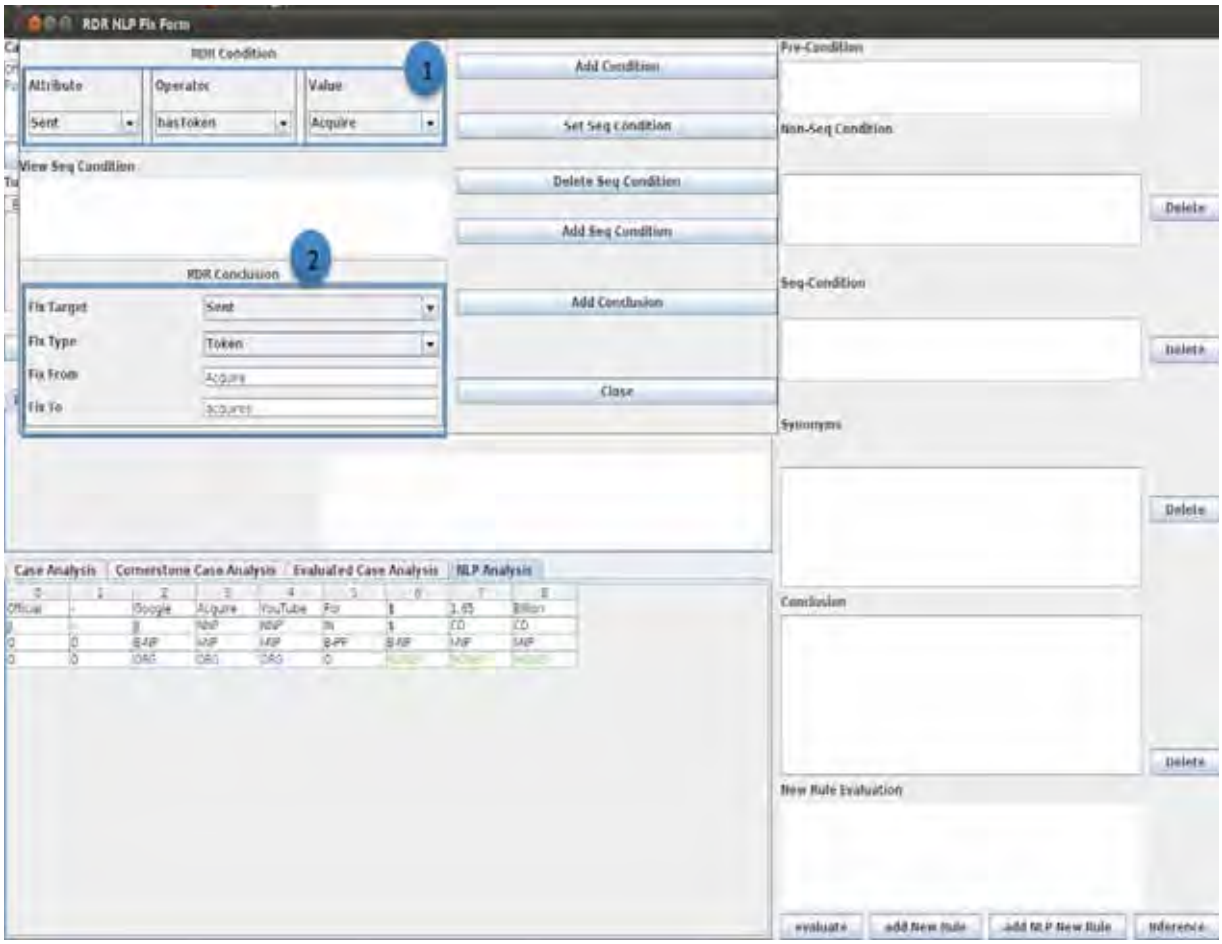

Figure 5.5. The KA process when creating a rule for the NLPRDR KB to refine the given case (a sentence)

Step1: The user sets the rule condition ‘IF (SENTENCE hasToken ‘**Acquire**’)’.

Step2: The user sets the rule conclusion ‘THEN (SENTENCE, Token, Acquire, acquires)’.

The rule is automatically checked for consistency with the existing knowledge base before it is committed.

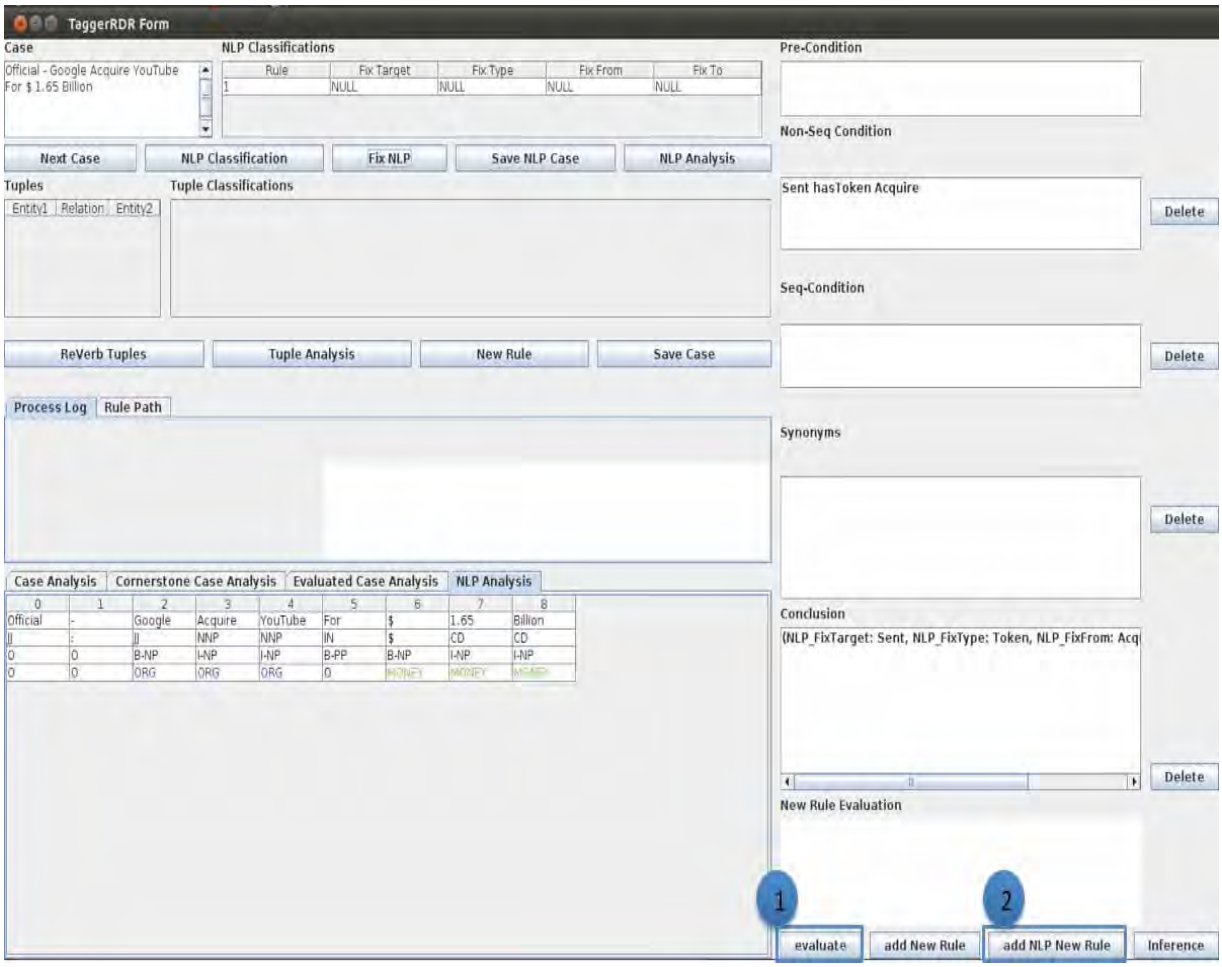

Figure 5.6. The KA process when entering a new rule into NLPRDR KB

Step 1: After setting the new rule R2's condition and conclusion, the user clicks the 'evaluate' button to check whether the new rule R2 under the default rule R1 conflicts with the existing NLPRDR KB.

Step 2: As no conflict occurs, the user clicks the 'add NLP New Rule' button to save the new rule R2 under the default rule R1 into NLPRDR KB.

The user checks that the case (sentence) is refined by R2 and then a tuple is extracted from the REVERB system correctly.

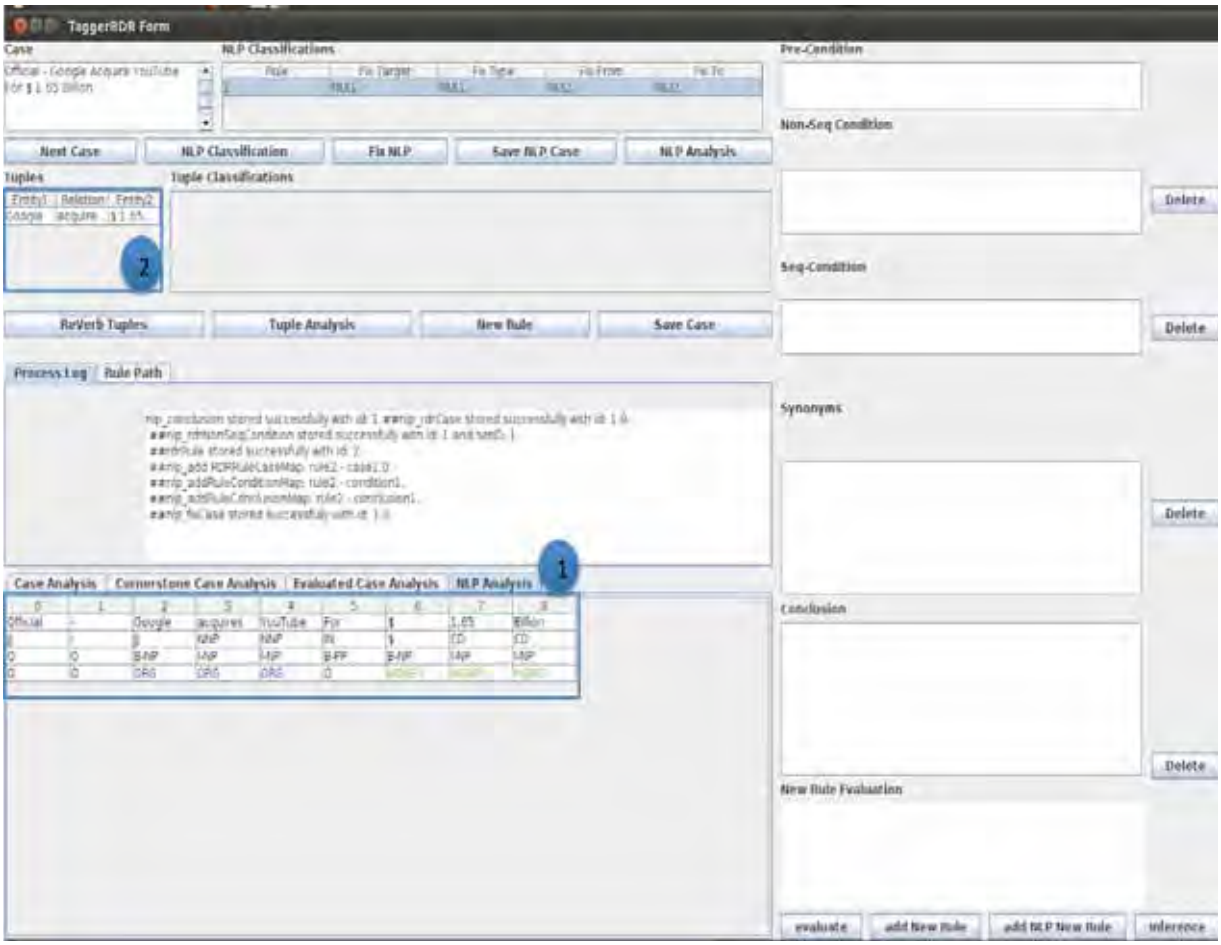

Figure 5.7. The KA process after entering a new rule R2 into NLPRDR KB

Step1: After saving the new rule R2, the user clicks the ‘NLP Analysis’ button and checks the sentence is refined correctly.

Step2: The user clicks the ‘REVERB Tuples’ button and gets one tuple extraction from the REVERB system.

The user gets a classification result from the TupleRDR KB and decides whether the tuple extraction is corret or needs to be refined by a rule.

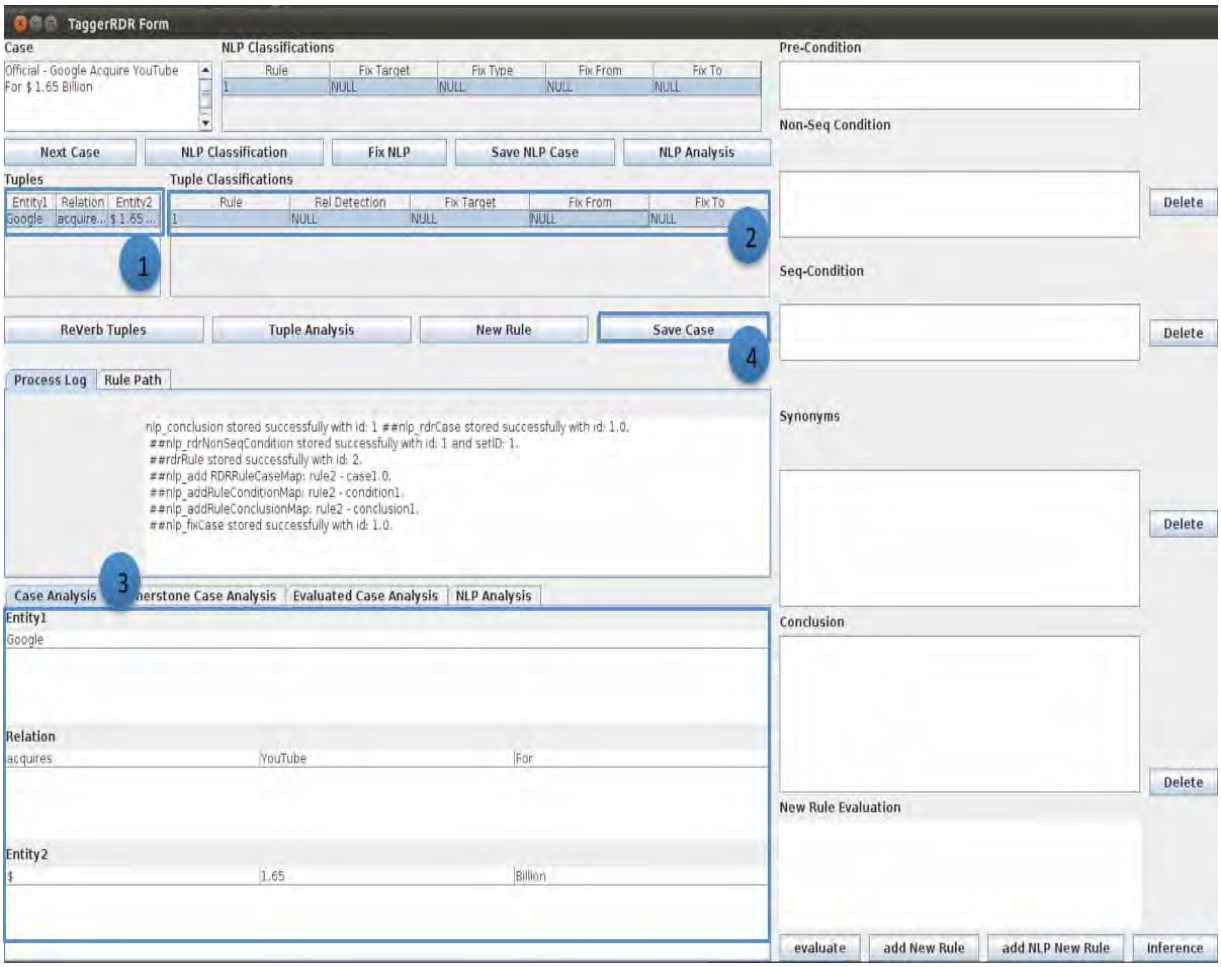

Figure 5.8. The KA process when getting a tuple extraction from the REVERB system

Step1: The user clicks a tuple extracted from the REVERB system.

Step2: The system automatically displays the classification result from the TupleRDR KB. The default rule R1 is fired and return a NULL classification result.

Step3: The user clicks the ‘Tuple Analysis’ button and checks each tuple elements, ENTITY1, RELATION and ENTITY2.

Step4: As the extracted tuple from the REVERB system (**Google** , **acquires YouTube for** , **$ 1.65 Billion**) is correct, the user saves the case (tuple) in the DB.

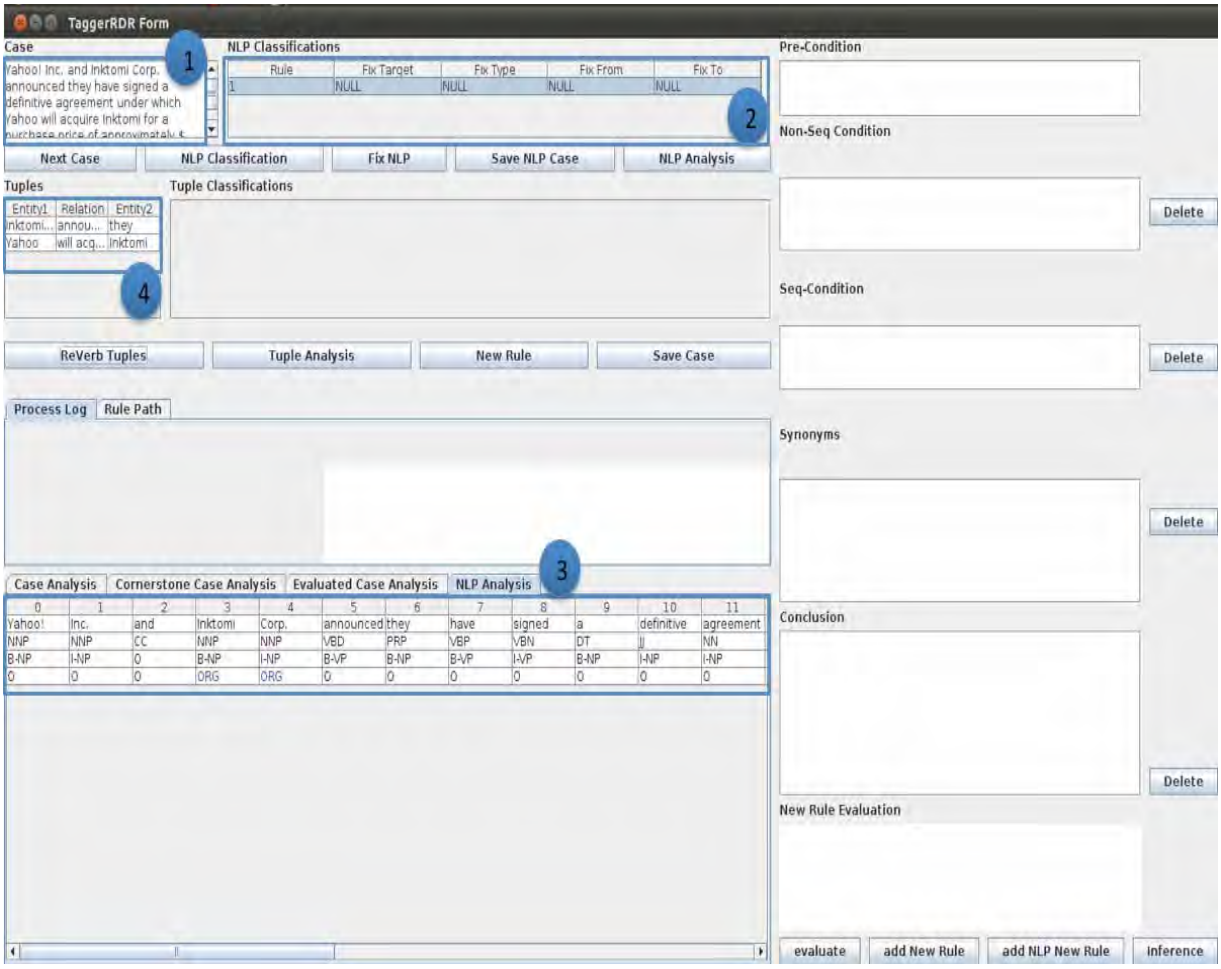

Figure 5.9. The KA process when a case (a sentence) is entered and two relation extractions (tuples) are produced from the REVERB system

Step1: The given sentence '*Yahoo! Inc. and Inktomi Corp. announced they have signed a definitive agreement under which Yahoo will acquire Inktomi for a purchase price of approximately $ 235 million.*' is displayed.

Step2: The user clicks the 'NLP Classification' button then the system automatically returns the classification result (the default rule R1 is fired and returns a NULL classification) from the NLPRDR KB.

Step3: Since the NLPRDR KB returns the NULL classification result, the user decides to check the features of the given sentence. The user clicks the ‘NLP Analysis’ button then the system displays NLP features such as part-of-speech tag, noun/verb chunk tag and named entity tag for each token of the sentence. The user finds that there is no NLP error to fix.

Step4: The user clicks the ‘REVERB tuples’ button. The REVERB system returns two tuples such as (**Inktomi Corp.** , **announced** , **they**) and (**Yahoo** , **will acquire** , **inktomi**).

The user gets a classification result from the TupleRDR KB and decides whether the tuple is correct or needs to be refined by a rule.

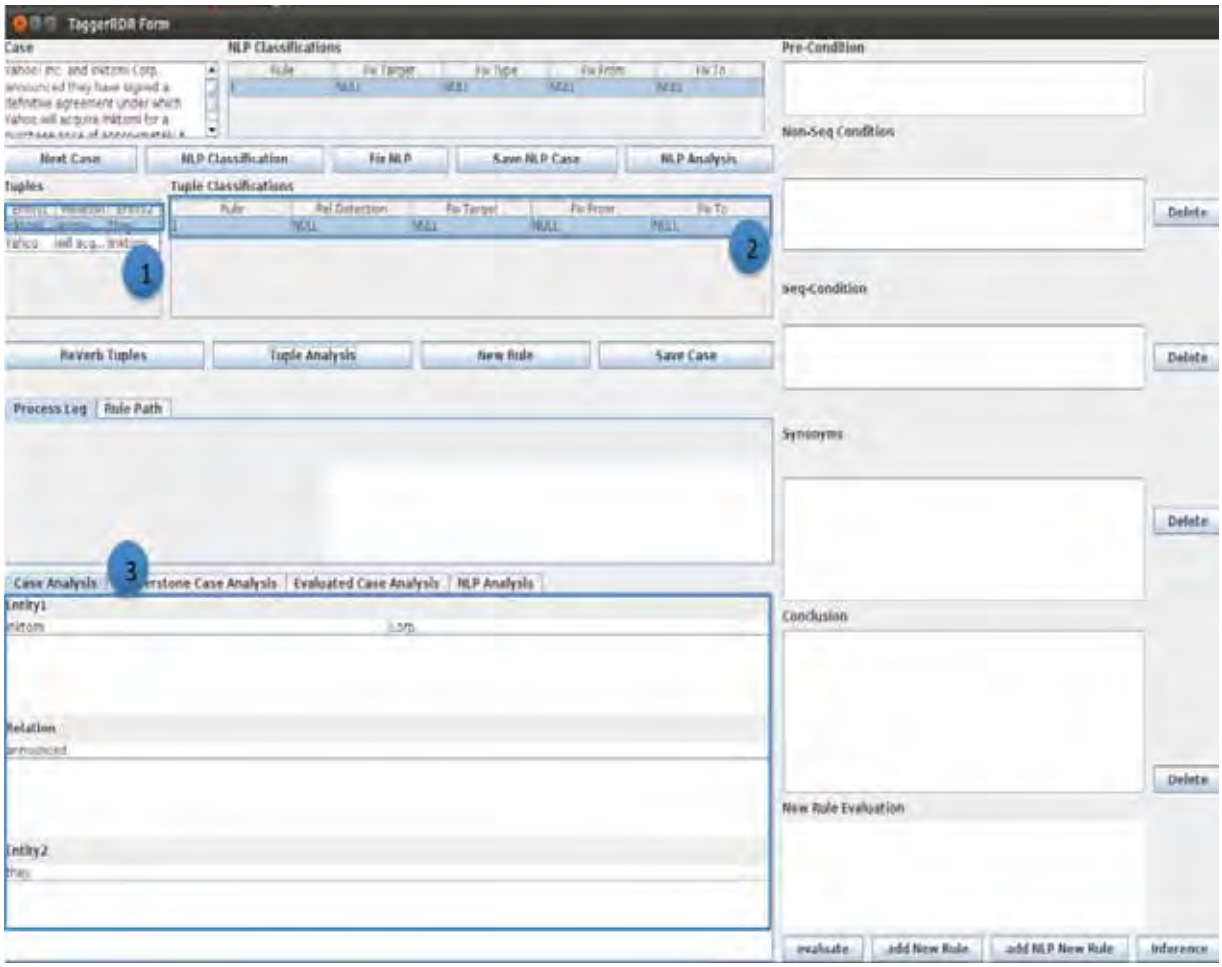

Figure 5.10. The KA process after getting tuples from the REVERB system

Step1: The user clicks the first tuple extracted from the REVERB system.

Step2: The system automatically displays the classification result from the TupleRDR KB. The default rule R1 is fired with a NULL classification result.

Step3: The user clicks the ‘Tuple Analysis’ button and checks each tuple elements, ENTITY1, RELATION and ENTITY2.

Step4: The user decides the extracted tuple (**Inktomi Corp.** , **announced** , **they**) is incorrect, and decides to create a rule to correct the classification result.

After the user clicks the ‘New Rule’ button on the main user interface, a small window pops up for a rule creation.

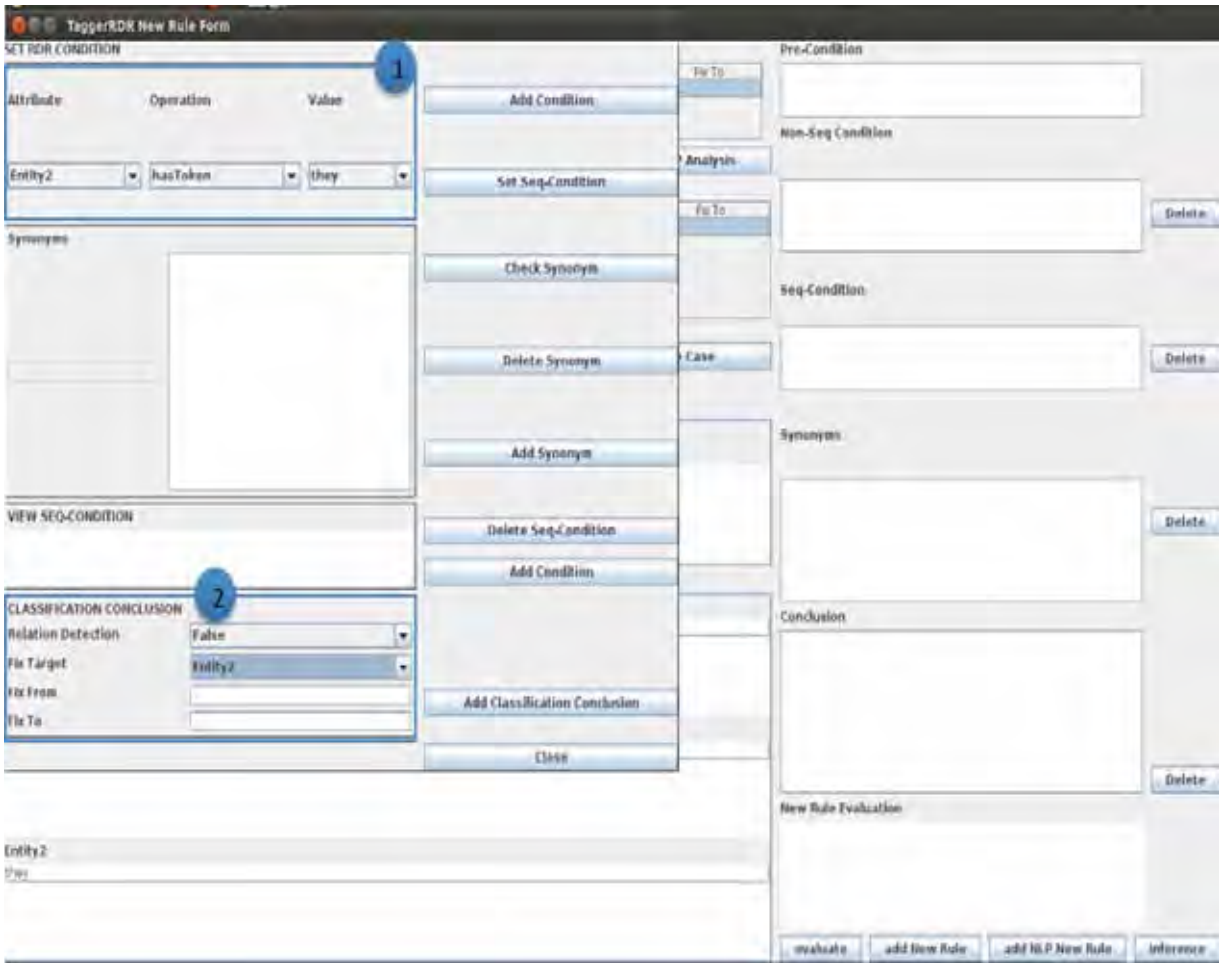

Figure 5.11. The KA process when creating a rule for the TupleRDR KB to correct the classification result

Step1: The user sets the rule condition ‘IF (ENTITY2 hasToken ‘**they**’)’.

Step2: The user sets the rule conclusion ‘THEN (False, ENTITY2, NULL, NULL)’.

The newly created rule is automatically checked for consistency with the existing knowledge base before it gets committed.

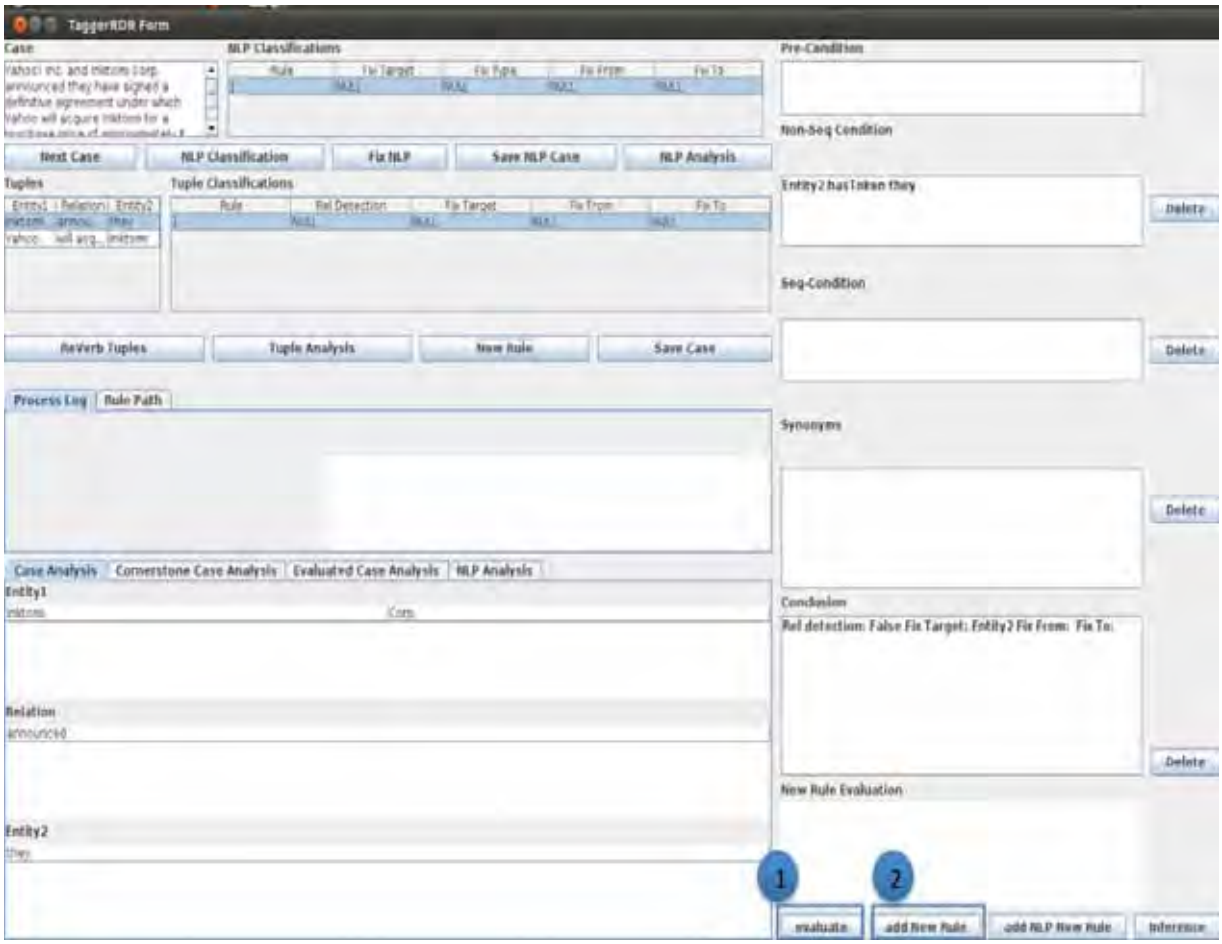

Figure 5.12. The KA process when entering a new rule into TupleRDR KB

Step 1: After setting the new rule R2's condition and conclusion, the user clicks the 'evaluate' button to check whether the new rule R2 under the default rule R1 conflicts with the existing TupleRDR KB.

Step 2: As no conflict occurs, the user clicks 'add NLP New Rule' button to save the new rule R2 into TupleRDR KB.

The user gets a classification result from the TupleRDR KB and decides whether the tuple is correct or needs to be refined by a rule. If the extracted tuple is the correct relation, then the user saves it in DB.

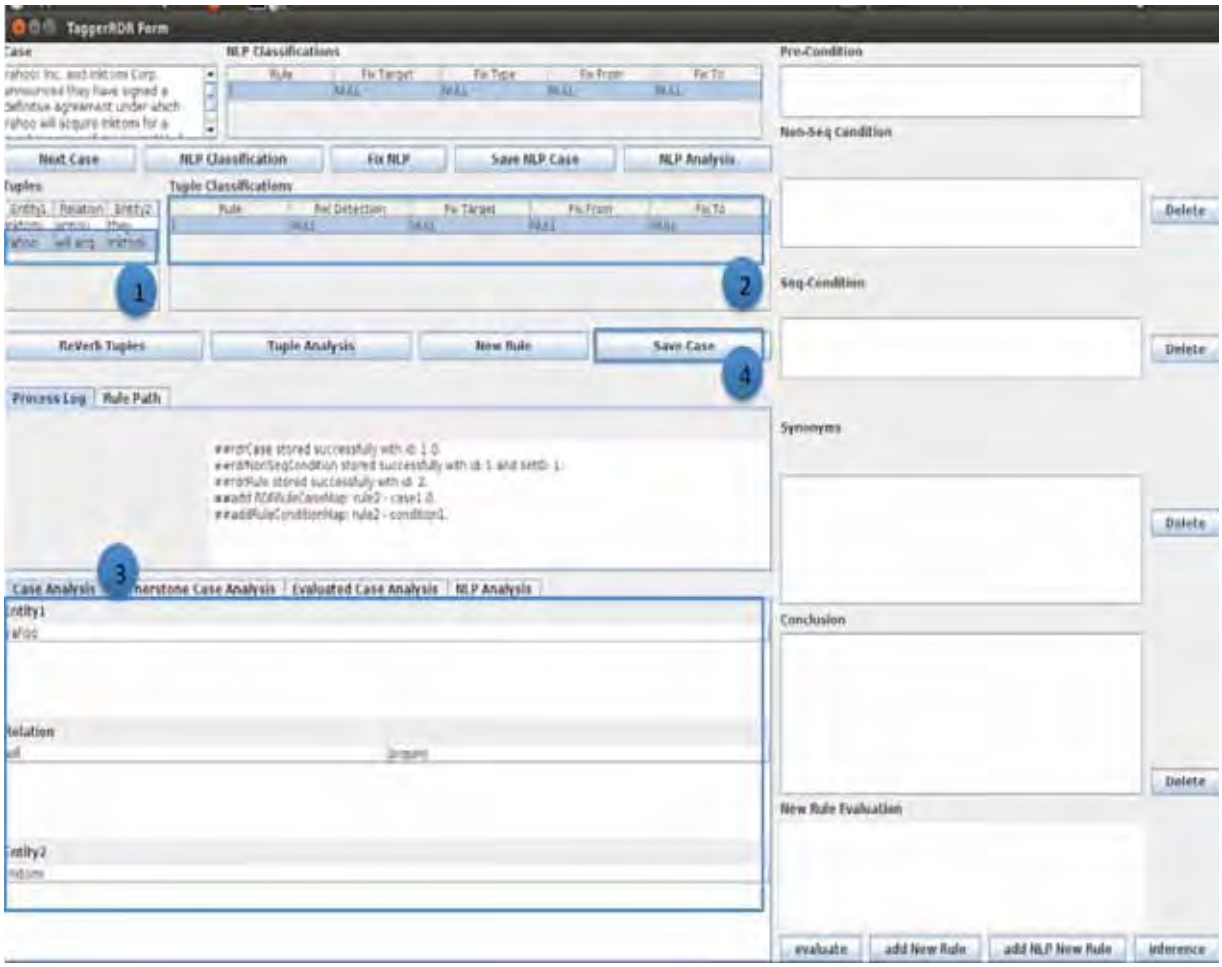

Figure 5.13. The KA process after getting tuples from the REVERB system

Step1: The user clicks the second tuple extracted from the REVERB system.

Step2: The system automatically displays the classification result from the TupleRDR KB. The default rule R1 is fired with a NULL classification result.

Step3: The user clicks the ‘Tuple Analysis’ button and checks each tuple element, ENTITY1, RELATION and ENTITY2.

Step4: As the extracted tuple (**Yahoo** , **will acquire** , **Inktomi**) is correctly extracted, the user saves the case (tuple) in the DB.

5.5 Experimental Setup

The experiments were conducted on the two Web datasets, Sent500 and Sent300, which were also used in the experiments for the RDROIE system. The Sent300 was used as a training dataset to construct the NLPRDR KB and TupleRDR KB and the Sent500 was used as a test dataset to test the performance of the overall Hybrid RDROIE system. As explained in section 4.4, the Sent300 is derived from the MIL dataset developed by Bunescu and Mooney (2007). The MIL dataset contains a bag of sentences which were collected from the Google search engine by submitting a query string ‘a1 ******* a2’ containing seven wildcard symbols between the given pair of arguments mainly to extract ‘corporate acquisition’ and ‘personal-birthplace’ relations. The Sent500 was developed by Banko and Etzioni (2008). 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. The Sent300 and the Sent500 datasets are described in depth in section 4.4. 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 for the open information extraction task. As well multiple tuples per sentence were extracted from the REVERB system while the study in chapter 4 described a system extracting single tuples based on pre-defined tags.

5.6 Knowledge Base (KB) Construction

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 is occurred. For the NLPRDR KB, in total, 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 single process flow. In total 53 rules were added within 2 hours. KB construction time covers from when a case is called up until a rule is accepted as complete and is logged automatically.

5.7 Hybrid RDROIE Performance

After building two initial KBs (the NLPRDR KB and the TupleRDR KB) with the Sent300, the Hybrid RDROIE system was tested on the Sent500 dataset. Table 5.4 presents the performance of the Hybrid RDROIE system on all extractions and on the four extraction categories. As briefly mentioned in section 5.1, the REVERB system extracts multiple tuples from a sentence without using pre-defined entity tags. The performance on all extractions is evaluated on all tuple

extractions of the REVERB system. The performance on four categories is calculated based on the explicit tuples when the pre-defined entity tags exist.

	ALL	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%

Table 5.4. The performance of the Hybrid RDROIE system on all extractions and on four categories of extraction on the Sent500 dataset

On all extractions, overall the Hybrid RDROIE system achieved 90% precision, 81.45% recall and 85.51% F1 score, while the REVERB system by itself achieved 41.32% precision, 45.25% recall and 43.20% F1 score on the same dataset (see table 5.1). That is, the Hybrid RDROIE system improved the performance of the REVERB system almost double on the Web dataset. Precision improved as the TupleRDR KB reduced false positive errors by filtering incorrect extractions and recall improved as the NLPRDR KB decreased false negative errors by amending informally written Web sentences.

Across the four categories extractions, on average the Hybrid RDROIE system improved around 30% for 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.

Category		VERB	NOUN+PREP	VERB+PREP	INFINITIVE
TEXTRUNNER	P	0.94	**0.89**	**0.95**	**0.96**
	R	0.65	0.36	0.50	0.47
	F1	0.77	0.51	0.66	0.63
heStateSnowball	P	**0.96**	0.70	***0.58***	0.46
	R	0.81	0.50	***0.34***	***0.19***
	F1	0.88	0.54	***0.40***	***0.23***
l1StatSnowball	P	0.93	0.82	0.62	***0.42***
	R	**0.85**	**0.78**	0.58	0.32
	F1	**0.89**	**0.80**	0.60	0.36
REVERB	P	0.70	***0.42***	0.70	0.50
	R	***0.56***	***0.26***	0.54	0.20
	F1	***0.62***	***0.32***	0.61	0.29
Hybrid RDROIE	P	***0.90***	0.74	0.90	0.85
	R	0.83	0.67	**0.86**	**0.77**
	F1	0.87	0.70	**0.88**	**0.81**

Table 5.5. Evaluation result of different OIE systems on four categories extractions on Sent500 dataset (the best results are shown in bold and the worst result are shown in bold and italics)

Table 5.5 shows the experimental results for previous machine learning-based general OIE systems, the REVERB system and our Hybrid RDROIE system on four categories extractions on Sent500 dataset, where the results of TEXTRUNNER and StatSnowball systems are from the original paper comparing the two systems (Zhu et al., 2009).

Overall, the Hybrid RDROIE system achieved good and balanced recall over four categories compared to other OIE systems. This is because RDR's incremental learning in the Hybrid RDROIE system successfully handled Web's informality while other OIE systems have focused more on the Web's heterogeneity than its informality. Also, the Hybrid RDROIE system demonstrates the most balanced performance (none of categories are under 50%) over the four categories.

One interesting point here is that overall the REVERB system performed worse than the TEXTRUNNER system over four categories for this data while the REVERB system outperformed the TEXTRUNNER system on other web datasets with less informality (Fader et al., 2011). This suggests that improving the REVERB system by handling the Web's informality is an important OIE task for Web text.

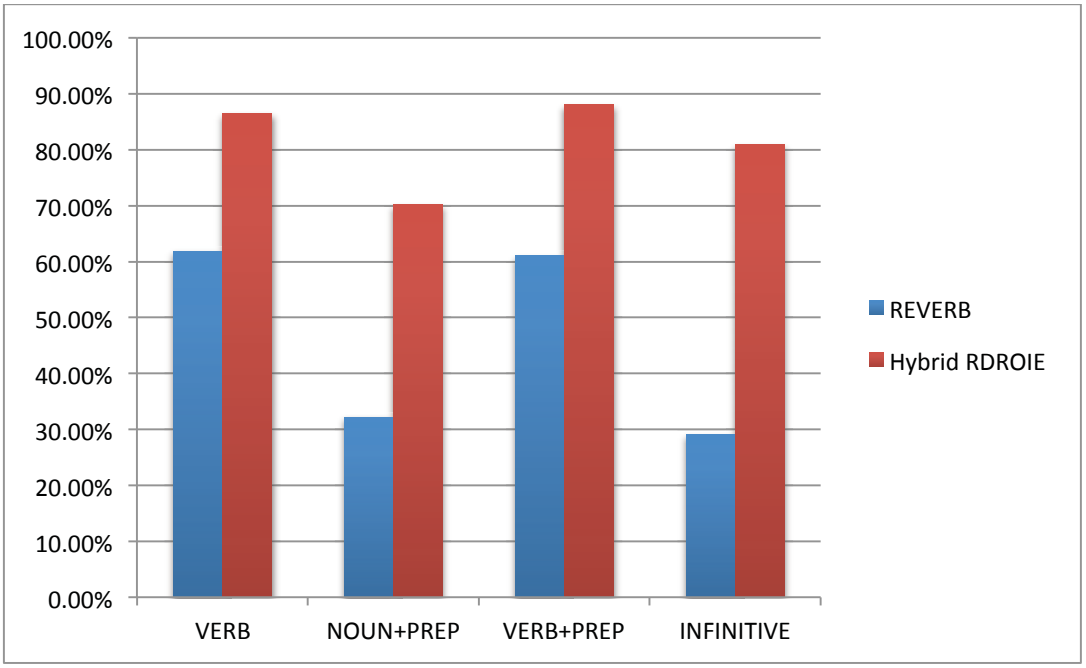

Figure 5.14. Performance improvement of the Hybrid RDROIE system from the REVERB system on the F1 score over four categories

Figure 5.14 presents the performance improvement of the Hybrid RDROIE system over the REVERB system on the F1 score over four categories. For all categories the Hybrid RDROIE system improved REVERB performance. In particular, NOUN+PREP and INFINITIVE categories had the biggest improvement.

5.8 How the Hybrid RDROIE System Handles the REVERB System Errors

In this section, we show how the Hybrid RDROIE system deals with the various types of errors: missed entity detection, entity boundary detection error, NLP error, unusual expression, non-verb relations (NOUN+PREP and INFINITIVE) which were discussed in section 5.1.

The following is an example of missed entity extraction. The entity 'YouTube' is not correctly extracted in the tuple. A simple rule is added into the TupleRDR KB to refine ENTITY2.

Error type	Missed entity extraction
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 __ .*'
Tuple extracted	(**Google** , **has acquired** , **the Video**)
RDR rule in TupleRDR KB	IF ((ENTITY1 hasToken '**Google**') & (RELATION hasToken '**acquired**') & (SENTENCE hasToken '**YouTube**')) Then (TRUE, ENTITY2, the Video, YouTube)
Resolved tuple	(**Google** , **has acquired** , **YouTube**)

The following is an instance of an entity boundary detection error. In the ENTITY1 element, 'Lee' is extracted instead of 'Tim Berners – Lee'. A rule is added into the TupleRDR KB to refine the ENTITY1.

Error type	Entity boundary detection error
Sentence	'*Tim Berners - Lee is the primary inventor of the World Wide Web , the system of text links and multimedia capabilities that made the Internet accessible to mass audiences* .'
Tuple extracted	(**Lee** , **is the primary inventor of** , **the World Wide Web**)
RDR rule in TupleRDR KB	IF ((ENTITY1 hasToken '**Lee**') & (SENTENCE hasToken '**Tim Berners - Lee**')) & THEN (TRUE, ENTITY1, Lee, Tim Berners - Lee)
Resolved tuple	(**Tim Berners – Lee** , **is the primary inventor of** , **the World Wide Web**)

The following is an example of NLP error due to informal capital letter usage. Because the word 'Buys' starts with a capital letter informally, the NLP tools used in the system tagged it as noun rather than verb. It was annotated 'NNP' for the POS tag and 'I-NP' for the chunk tag. Thus, no tuple is extracted from the REVERB system. A rule is added into the NLPRDR KB to refine the token 'Buys' as 'buys'.

Error type	NLP error – Informal capital letter usage
Sentence	'*Update 2 : Novartis Buys Drugmakers Hexal and Eon*'
Tuple extracted	None
RDR rule in NLPRDR KB	IF (SENTENCE hasToken '**Buys**') THEN (SENTENCE, Token, Buys, buys)
Resolved tuple	(**Novartis** , **buys** , **Drugmakers Hexal and Eon**)

The following is an instance of an NLP error due to misspelling. Because the word 'aquires' is misspelled, the NLP tools used in the system tagged it as noun rather than verb. It was annotated 'NNS' for the POS tag and 'I-NP' for the chunk tag. Thus, no tuple is extracted from the REVERB system. A rule is added into the NLPRDR KB to refine the token 'aquires' as 'acquires'.

Error type	NLP error – Misspelling
Sentence	'*Google aquires Youtube = $ 1.65Billion*'
Tuple extracted	None
RDR rule in NLPRDR KB	IF (SENTENCE hasToken '**aquires**') THEN (SENTENCE, Token, aquires, acquires)
Resolved tuple	(**Google** , **acquires** , **Youtube**)

The following is an example of error due to an unusual expressions in the dictionary of the REVERB system. As explained in section 5.1, the REVERB system contains approximately 1.7 million distinct normalised relation phrases, which were derived from 500 million Web sentences. The system uses this relation phrase dictionary to detect relations, and thus tends to miss relations not included in the dictionary. Because the relation 'born' is not a usual expression in the dictionary, the REVERB system extracted no tuples from the given sentence. A rule is added into the NLPRDR KB to amend the token 'born' as 'was born'.

Error type	Unusual expression
Sentence	'*Franz Kafka born July 3 , 1883 , Prague .*'
Tuple extracted	None
RDR rule in NLPRDR KB	IF (SENTENCE hasToken '**born**') THEN (SENTENCE, Token, born, was born)
Resolved tuple	(**Franz Kafka** , **was born, July 3** , **1883**)

The following is an instance of error due to non-verb relations, especially the NOUN+PREP category. As explained in section 5.1, the REVERB system aims to extract binary relations expressed mainly by verbs. Thus, the system cannot extract NOUN+PREP category relations unless a verb exists in the RELATION element like (Entity1, verb NOUN+PREP, Entity2). In order to extract the NOUN+PREP relation category when no verb exists in the RELATION

element, a simple rule is added to convert the non-verb relation phrase ‘acquisition in’ as ‘acquires’.

Error type	Non-verb relation – NOUN+PREP type
Sentence	‘*Here is the video of the two _ founders talking about the Google acquisition in their YouTube Way !*’
Tuple extracted	None
RDR rule in NLPRDR KB	IF SEQ((SENTENCE hasToken ‘**acquisition**’) & (SENTENCE hasToken ‘**in**’)) THEN (SENTENCE, Token, acquisition in, acquires)
Resolved tuple	(**Google** , **acquires** , **YouTube**)

The following is an instance of error due to non-verb relations, in particular the INFINITIVE category. As explained in section 5.1, the REVERB system aims to extract binary relations expressed mainly by verbs. Thus, the system cannot extract INFINITIVE category relations unless a verb exists in the RELATION element like (Entity1, verb TO VB, Entity2). In order to extract then INFINITIVE relation category when no verb exists in the RELATION element, a simple rule is added to convert the non-verb relation phrase ‘to Buy’ as ‘buys’.

Error type	Non-verb relation – INFINITIVE type
Sentence	'*Adobe About to Buy Macromedia .*'
Tuple extracted	None
RDR rule in NLPRDR KB	IF SEQ((SENTENCE hasToken '**to**') & (SENTENCE hasToken '**Buy**')) THEN (SENTENCE, Token, to Buy, buys)
Resolved tuple	(**Adobe** , **buys** , **Macromedia**)

5.9 Discussion

Table 5.4 shows that the Hybrid RDROIE system achieves high precision and recall after only 2 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 on the informal Web dataset, it is a worthwhile and practical alternative to very rapidly add rules to specifically cover the REVERB system's degraded performance on informal Web text in a specific domain.

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

The advantage of utilising lexical features directly was demonstrated by the REVERB system's outstanding performance compared to previous OIE systems such as the TEXTRUNNER system and the WOE system (Banko and Etzioni, 2008; Wu and Weld, 2010), but we note that this was for data without a high level of informality as occurs in the data here. The REVERB system primarily utilises direct lexical feature matching techniques using the relation phrase dictionary, collected from 500 million Web sentences. On the other hand, previous OIE systems such as the TEXTRUNNER system and the WOE system utilise more NLP features like part-of-speech and noun/verb chunk phrase and use machine learning techniques on a large number of heuristically labelled training dataset (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 improves the REVERB system's performance further. In consequence, as shown in table 5.5, the Hybrid RDROIE system outperformed the REVERB system and achieved a good and balanced overall result compared to other OIE systems. Section 5.8 showed examples of the Hybrid RDROIE system using lexical features in rule creation. We note that the other systems focusing more on NLP issues outperformed the REVERB system on this data set, but we also note that as shown in chapter 4 a pure RDR approach did even better.

The Hybrid RDROIE system is designed to be trained on a specific domain of interest. We have not yet explored whether the rules for new domains should be added to rules for a previous domain, or whether it is better to start again (but again wrapping the rule system around a general OIE system like the REVERB system). One might also comment that the rules added are not very interesting and tend to be very simple fixes of lexical errors, and to produce a large system would probably need a very large number of such rules. We note that this is really the same type

of approach as the REVERB system's approach 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 built systems with over 10,000 rules (Compton et al 2011). The Hybrid RDROIE system required very little work; a rule creation is very simple and rapid. In 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 (Compton et al., 2011). 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 version of crowd sourcing, whereby large numbers of people on the web might contribute rules.

Chapter 6. Ripple-Down Rules based Named Entity Recognition (RDRNER)

In chapter 5, we introduced the idea of RDR on top of a general system in the Hybrid RDROIE system and demonstrated the advantage of RDR on top of the REVERB system. In this chapter, we investigated this approach for a Web-scale named entity recognition task.

In this chapter, we describe the RDRNER (Ripple-Down Rules based Named Entity Recognition) system. The RDRNER system employs the Stanford NER system as a base NER system and applies the Ripple-Down Rules technique to deal with the Web's informality in a specific domain. This is an important separate task because in building the Hybrid RDROIE we used the Stanford NER system to identify named entities, as we had previous in the RDROIE system described in chapter 4. Since the Stanford NER system itself is error prone on informal

Web text, we use the same idea to clean up its errors using RDR's incremental learning. A fully comprehensive hybrid system would include fixing NER errors as part of the overall process, here we discuss fixing NER errors as a separate task.

The underlying idea of the RDRNER system is that it takes the state-of-the-art performance from machine learning techniques but then corrects any errors due to the Web's informality. In the RDRNER system, the user creates rules when the classification result provided by the base system is incorrect. As the rules are generated by humans, the RDRNER system is more likely to be able to handle NER errors caused by informally written Web documents.

Section 6.1 illustrates how the Web's informality challenges existing the state-of-the-art NER system and section 6.2 describes the architecture of the RDRNER system. Section 6.3 presents the description of RDR rule and examples of MCRDR structure in the RDRNER system. Section 6.4 demonstrates the knowledge acquisition process of the RDRNER system through screenshots. Section 6.5 presents the experiments to compare the performance of three widely used NER systems: the machine learning-based Stanford NER system, the heavy web gazetteer based LBJ NER system and the rule-based GATE NER system on Wikipedia documents. The typical types of errors and their causes are discussed for each NER system. Section 6.6 presents the experimental setup and section 6.7 describes the initial knowledge base (KB) construction of the RDRNER system. Section 6.8 presents the improved performance from the base system, the Stanford NER, to our RDRNER system and describes how the RDRNER system handles the Web's informality. Lastly, section 6.9 discusses the results.

6.1 The Web's Informality and NER

In this section, we illustrate the performance of the Stanford NER system on a Web dataset and categorise the types of NER errors due to the Web's informality. In the experiment the Stanford NER (version 1.5) that was used for our previous RDROIE systems is tested on Sent500.

	Person	Organisation	Location	Money	Date	Time	Percent	All
P	90.40%	86.19%	84.40%	97.22%	87.18%	100%	100%	87.10%
R	75.33%	62.53%	81.42%	85.37%	85.00%	100%	66.67%	69.23%
F1	81.82%	72.72%	82.47%	90.60%	85.99%	100%	80.24%	76.96%

Table 6.1. The performance of the Stanford NER system on the Web dataset, Sent500

Table 6.1 presents the performance of the Stanford NER system on seven NE classes in the Web dataset. The details of the Web dataset are discussed in section 6.5. CONLL evaluation methodologies were used for the experiment. Overall, the Stanford NER system achieved a 76.96% F1 score on the Web dataset, while it achieved a state-of-the-art 90.8% F1 score on the CONLL corpus (Ratinov and Roth, 2009). This is a 14% performance drop on informal Web documents compared to formal journalistic documents without noise. On average, the Stanford NER system achieved quite reasonable precision but low recall on informal Web documents. The difference between precision and recall is around 18%.

New vocabularies	39.53%
ML inconsistency	25.97%
Informal capital letter usage	21.71%
Lack of trigger words	10.08%
Web noise	2.71%

Table 6.2. The Stanford NER system's error sources on the Web dataset, Sent500

Table 6.2 presents the error causes of the Stanford NER system on the informal Web dataset. Error sources were classified into five categories: new vocabularies, Machine Learning (ML) inconsistency, informal capital letter usage, lack of trigger words and various Web noise.

39.53% of the errors were caused by the new vocabularies that had not been seen during training and were not contained in dictionaries used by the system. As noted in section 2.2.2, in order to solve this problem, some studies have focused on creating gazetteers automatically from the Web (Etzioni et al., 2005) or Wikipedia (Toral and Mu˜noz, 2006) to increase the coverage of the gazetteers. As the size of these Web gazetteers is quite large, most systems use a simple 'bag of words' approach to manage these large gazetteers, which can cause context analysis conflicts as their size increase. Another problem is the uncontrolled noise contained in the Web gazetteers. Liu et al. (2011) have shown that the noise contained in the Web gazetteers reduces NER system performance.

25.97% of the errors were caused by the Machine Learning (ML)'s inconsistent annotations. For example, in one short Web sentence, '(/**O** Encyclopedia/**ORG**)/**O** Kafka/**PER**', the word 'Kafka' is annotated as a PERSON NE. However, in another short sentence, '(/**O** Almanac/**ORG**

–/**O** People/**O**)/**O** Kafka/**LOC** ', the word 'Kafka' is tagged as a LOCATION NE although the second sentence has the better trigger word 'People'. It is difficult to understand why such errors are made.

21.71% of the errors were triggered by informal Web capital letter usage. On the Web, capital letters are often used informally in order to emphasise the contents, but this causes critical errors for NER systems trained on a formal corpus. For instance, in the sentence 'Google/**ORG** Acquires/**ORG** YouTube/**ORG**', the system annotated all three tokens as a single ORGANISATION NE tag instead of two ORGANISATION NE like 'Google/**ORG** Acquires/**O** YouTube/**ORG**' because the word 'Acquires' started with a capital letter informally.

10.08% of errors were caused by a lack of trigger words expected such as 'Ltd. and Dr.'. For example, in a sentence 'Franz/**PER** Kafka/**PER** –/**O** Prague/**LOC** wax/**O** museum/**O**', 'wax museum' was not tagged as LOCATION NE due to a lack of trigger words indicating location.

2.71% of errors were triggered by Web noise such as spelling errors, various symbols, excessive abbreviations and informal shorthand. For instance, in a sentence, 'This/**O** morning/**O** Googld/**O** held/**O** a/**O** webcast/**O** and/**O** conference/**O** call/**O** session/**O** with/**O** Eric/**PER** Schmidt/**PER** (/**O** Google/**ORG** CEO/**O**)/**O**', the word 'Googld' was not tagged as NE due to a spelling error.

6.2 RDRNER System Architecture

The RDR-based Named Entity Recognition (RDRNER) system shown in Figure 6.1 consists of three main components: preprocessor, Stanford NER system and RDR KB learner.

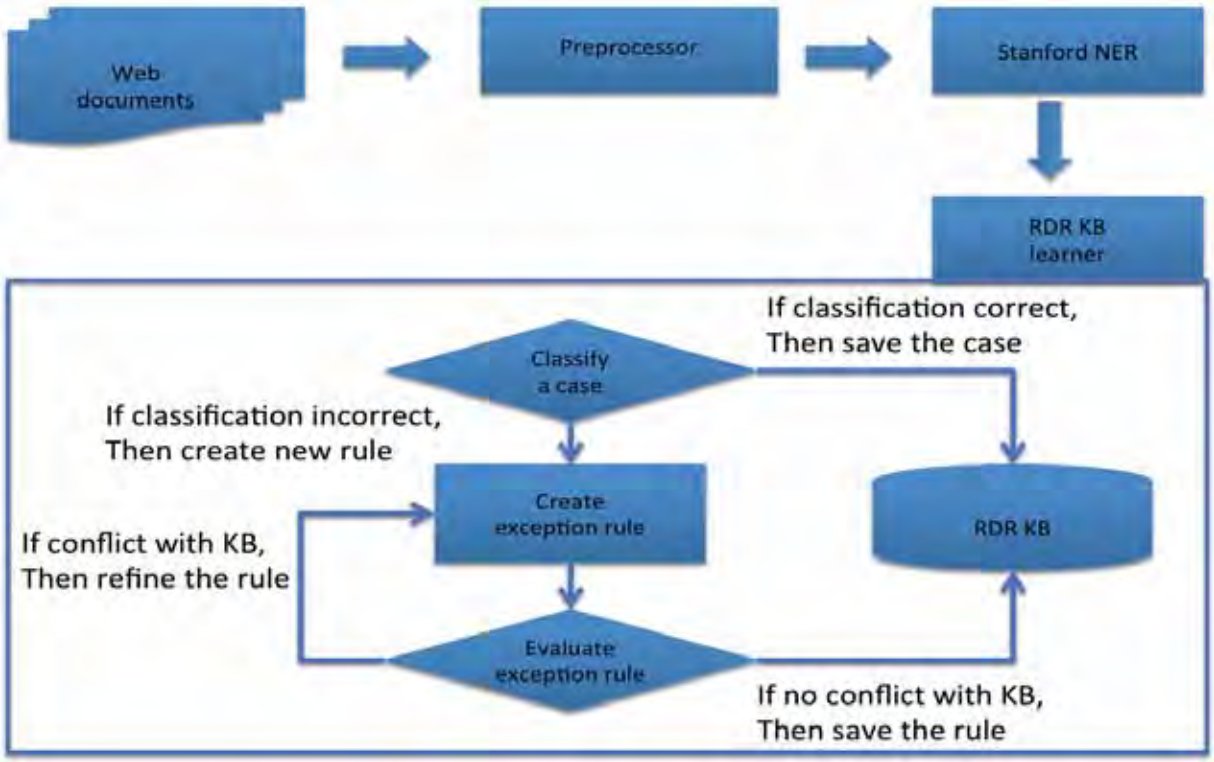

Figure 6.1. Architecture of the RDRNER System

6.2.1 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. Annotated NLP features are utilised when creating RDR rules.

6.2.2 Stanford NER System

The Stanford NER system was chosen as the base NER system because it is well known as one of the most robust NER systems. It also performed best on our experiment (with version 1.1. of the Stanford NER system) when compared with other three widely used NER systems on Wikipedia documents (see section 6.5). In particular we chose the Stanford NER classifier 'muc.7class.distsim.crf.ser.gz' which was trained on MUC corpora and classifies seven types of

NE classes: PERSON, ORGANISATION, LOCATION, DATE, TIME, MONEY, and PERCENT. The selected classifier used Gibbs sampling for inference in a Conditional Random Field (CRF) machine learning technique (Finkel et al., 2005) and Alexander Clark's distributional similarity code to improve performance (Clark and Tim, 2003), which has additional features providing better performance but require more memory.

6.2.3 RDR KB Learner

The RDRNER KB system is built incrementally while the system is in use with the base NER system. In the RDRNER system, the user gets the NER classification results from the Stanford NER system and adds rules when the classification results are not correct. There are three following steps.

Step1: RDRNER classification

The RDR KB takes the given preprocessed sentence and the NER classification results from the Stanford NER system then returns the RDRNER classification results. If RDR rules are fired, the system replaces the Stanford NER system results as the fired RDR rule's conclusion. Otherwise, the Stanford NER results are given.

Step2: Create RDR rule

Whenever incorrect classification results are given (by the Stanford NER or the RDR KB add-on), the user adds rules to correct the classification results.

Step3: Evaluate and refine the 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. As with the systems described in the previous chapters, the expert must select at least one differentiating feature, unless they decide that new conclusion should apply to the previous case.

6.3 RDRNER Rule Description

An RDR rule comprises of a condition part and a conclusion part. It has the form:

if CONDITION

then CONCLUSION

where CONDITION indicates more than one condition.

6.3.1 Condition

A CONDITION consists of three components: ATTRIBUTE, OPERATOR and VALUE.

6.3.1.1 Attribute

An ATTRIBUTE refers to each token of the given sentence.

6.3.1.2 Operator

The RDRNER system provides 8 types of OPERATOR as follows:

• 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 generate a pattern

• notHasPOS: whether a certain part-of-speech does not match

• notHasNE: whether a certain named entity does not match

• beforeWD(+a): checks tokens positioned before the given attribute token by +a

• afterWD(+a): checks tokens positioned after the given attribute token by +a

6.3.1.3 Value

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

6.3.1.4 Type of Conditions

Conditions are connected with an 'and' operation. A sequence of conditions begins with a 'SEQ' keyword and is used to identify a group of words in sequence order, so patterns can be detected.

For instance, the sequence condition:

'SEQ(('**Prague**' hasNE '**LOC**') & ('**wax**' hasNE '**O**') & ('**museum**' hasNE '**O**'))'

detects 'Prague/**LOC** wax/**O** museum/**O**'.

6.3.2 Conclusion

The rule's CONCLUSION part has the following form:

(fixTarget, --- target word

fixType, --- refinement type, default is NE in RDRNER system

fixFrom, --- classification result before refinement

fixTo) --- classification result after refinement

6.3.3 Examples of RDRNER Rules

The RDRNER system is based on Multiple Classification RDR (MCRDR) (Kang et al., 1995). Figure 6.2 demonstrates MCRDR based KB construction as the RDRNER system processes the following three cases starting on an empty KB (with a default rule R1 which is always true and returns a NULL classification).

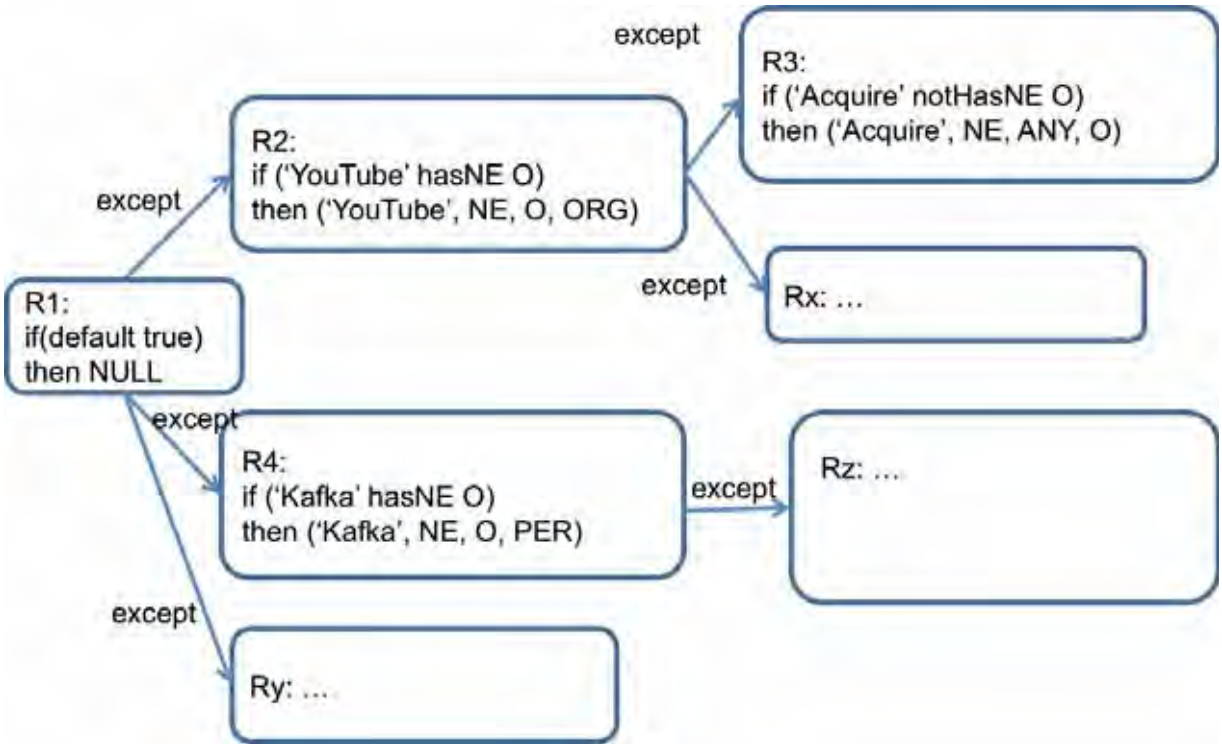

Figure 6.2. MCRDR Structure of the RDRNER System

Case1: ‘Since/**O** Google/**ORG** just/**O** bought/**O** YouTube/**O** today/**DATE** for/**O** 1.6/**MONEY** billion/**MONEY** dollars/**MONEY** in/**O** stock/**O’**

→ The default rule R1 is fired and the KB system returns a NULL classification, which the user decides is an incorrect classification result because ‘YouTube’ is not tagged as an ORGANISATION NE.

→ A user adds a new rule R2 under the default rule R1.

Case2: ‘One/**O** Response/**O** to/**O** Official/**O** -/**O** Google/**ORG** Acquire/**ORG** YouTube/**O** For/**O** $/**MONEY** 1.65/**MONEY** Billion/**MONEY’**

→ Rule R2 is fired but the user justifies this is as an incorrect result because ‘Google’ and ‘YouTube’ should be tagged as two separate ORGANISATION NEs instead of tagging ‘Google Acquire YouTube’ as one ORGANISATION NE.

→ A user adds an exception rule R3 under the parent rule R2.

Case3: ‘Kafka/**O** died/**O** from/**O** tuberculosis/**O** in/**O** 1924/**DATE** and/**O** is/**O** buried/**O** in/**O** the/**O** OI/**O** cemetery/**O** in/**O** Prague/**LOC** ./**O’**

→ The default rule R1 is fired and the KB system returns a NULL classification, which the user decides it as an incorrect result because ‘Kafka’ is not tagged as a PERSON NE.

→ A user adds new rule R4 under the default rule R1.

6.4 Knowledge Acquisition (KA) Process for the RDRNER System

In this section, we present a number of screenshots of the RDRNER system's user interface to demonstrate the Knowledge Acquisition (KA) process while handling three given cases (case 1 to case 3) in section 6.3.3 starting with an empty KB. In the RDRNER system, the user conducts the knowledge acquisition process through a user friendly interface, which automatically displays features needed in each KA process. On the following screenshots, each KA process is numbered to indicate its order and the description for each process step is presented below the screenshot. The RDRNER system is written in Java (Java 1.6) and it adopts the OpenNLP system (version 1.5) and the Stanford NER system (version 1.5). It is built on top of the RDROIE system as the aim is to support the tuple extractor of the RDROIE system.

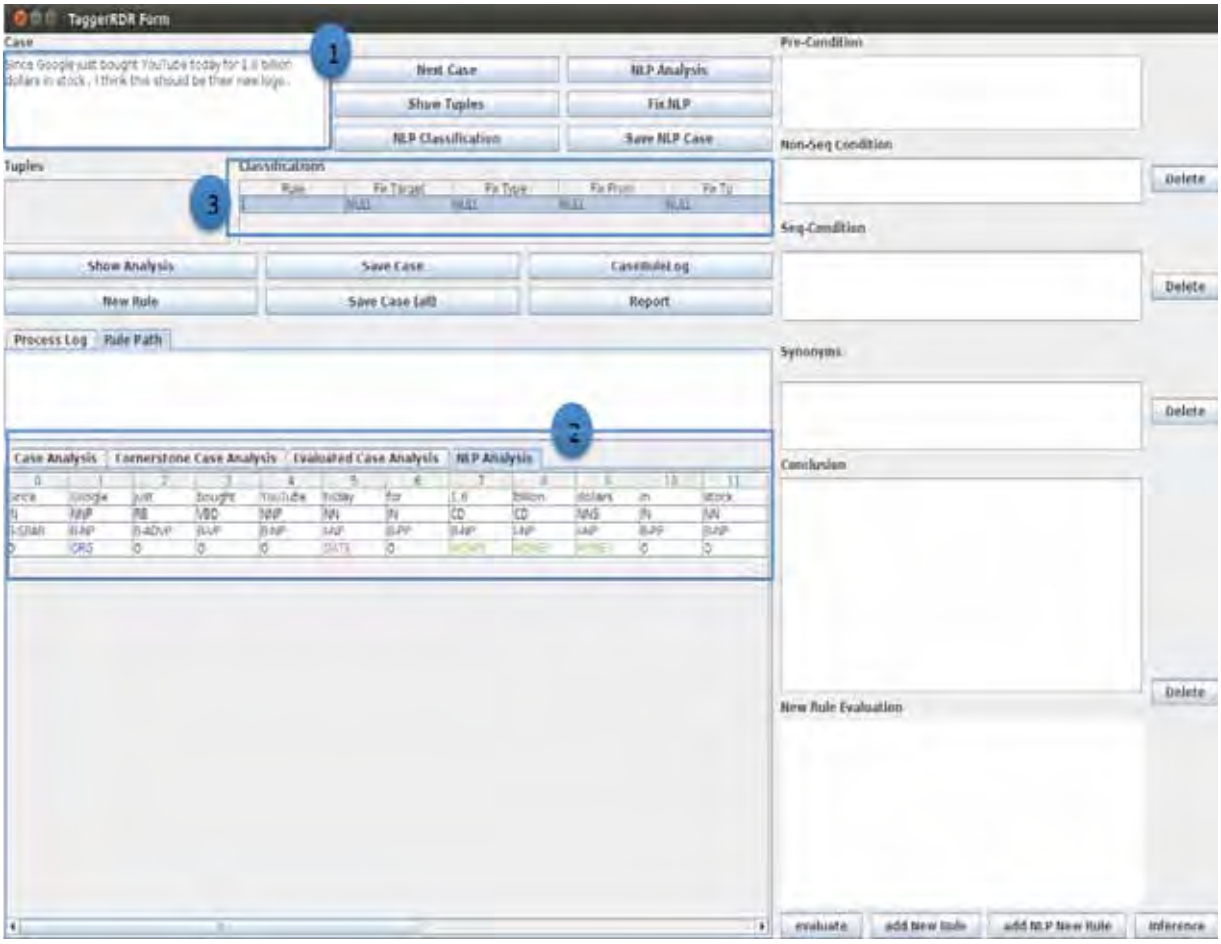

Figure 6.3. The KA process when a new case (a sentence) is entered into the system with an empty KB

Step 1: The given sentence '*Since Google bought YouTube today for 1.6 billion dollars in stock.*' is displayed.

Step 2: After clicking the 'NLP Analysis' button, the user views the 'NLP Analysis' table to check whether any NE errors occurred. The user notices that 'YouTube' is not tagged as an ORGANISATION NE.

Step 3: The user clicks the ‘NLP Classification’ button and the system automatically returns the NULL classification result from the KB (which is empty). Because no rule is fired to fix the error, the user decides to create a new RDR rule R2 under the default rule R1 to correct the NE error.

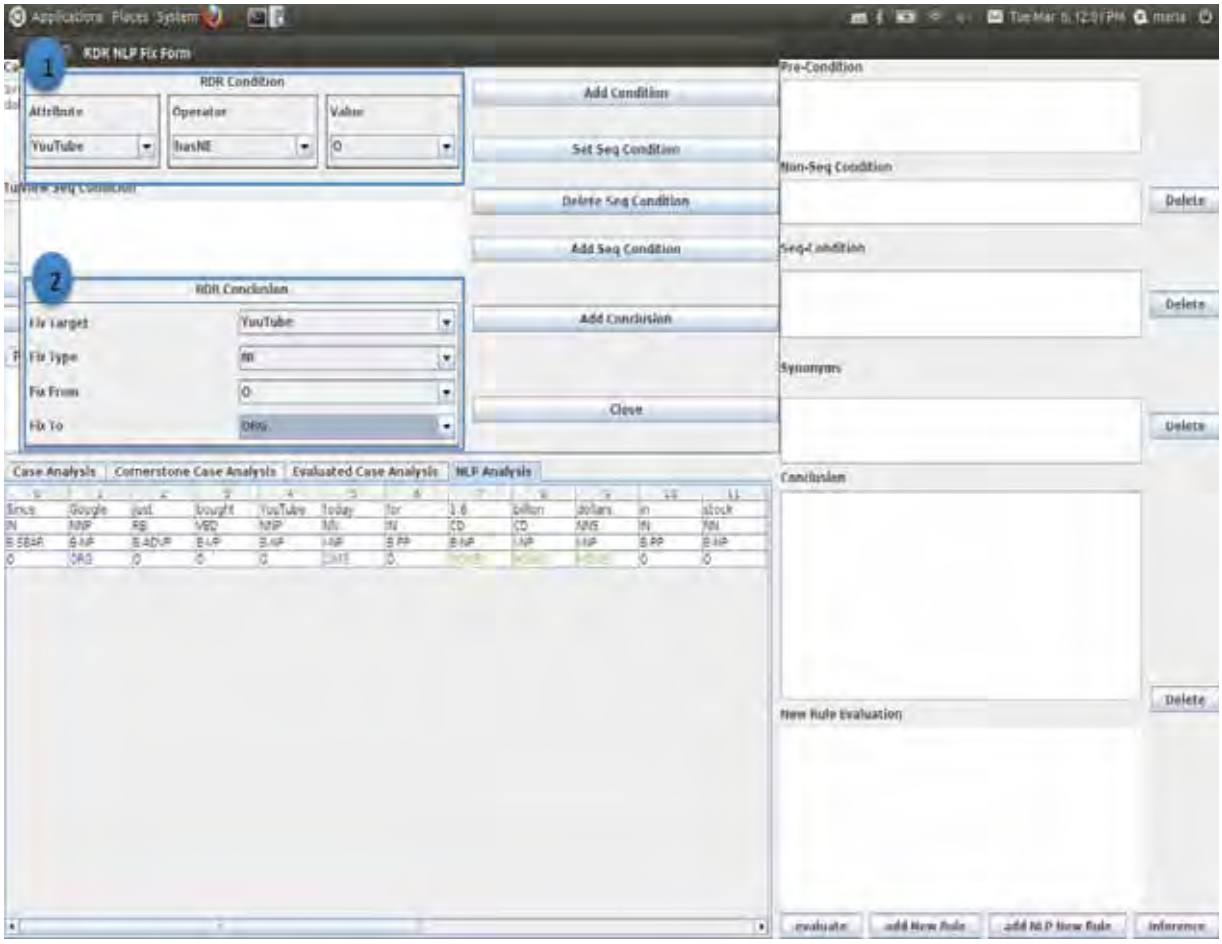

Figure 6.4. The KA process when the user clicks ‘fix NLP’ button to create a new rule and the ‘RDR New Rule Form’ window pops up

Step 1: The user sets the rule condition ‘IF(‘**YouTube**’ hasNE ‘**O**’)’.

Step 2: The user sets the rule conclusion THEN (YouTube, NE, O, ORG).

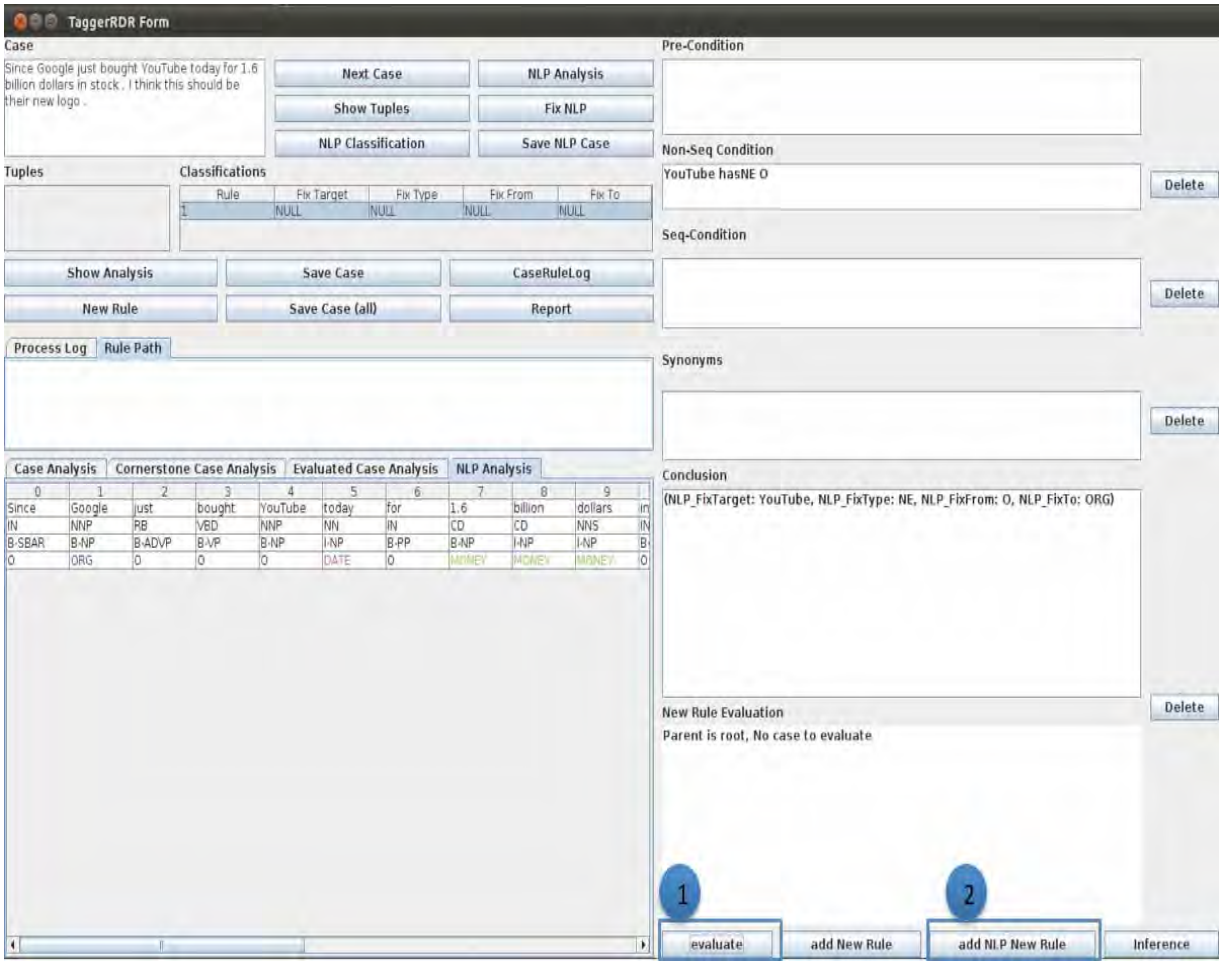

Figure 6.5. The KA process when evaluating the new rule that has been created and saving it into the KB system

Step 1: After creating the new rule R2's condition and conclusion, the user clicks the 'evaluate' button to check whether the new rule R2 under the default rule R1 conflicts with the existing KB.

Step 2: As no conflict occurs, the user clicks the 'add NLP New Rule' button to save the new rule R2 under the default rule R1.

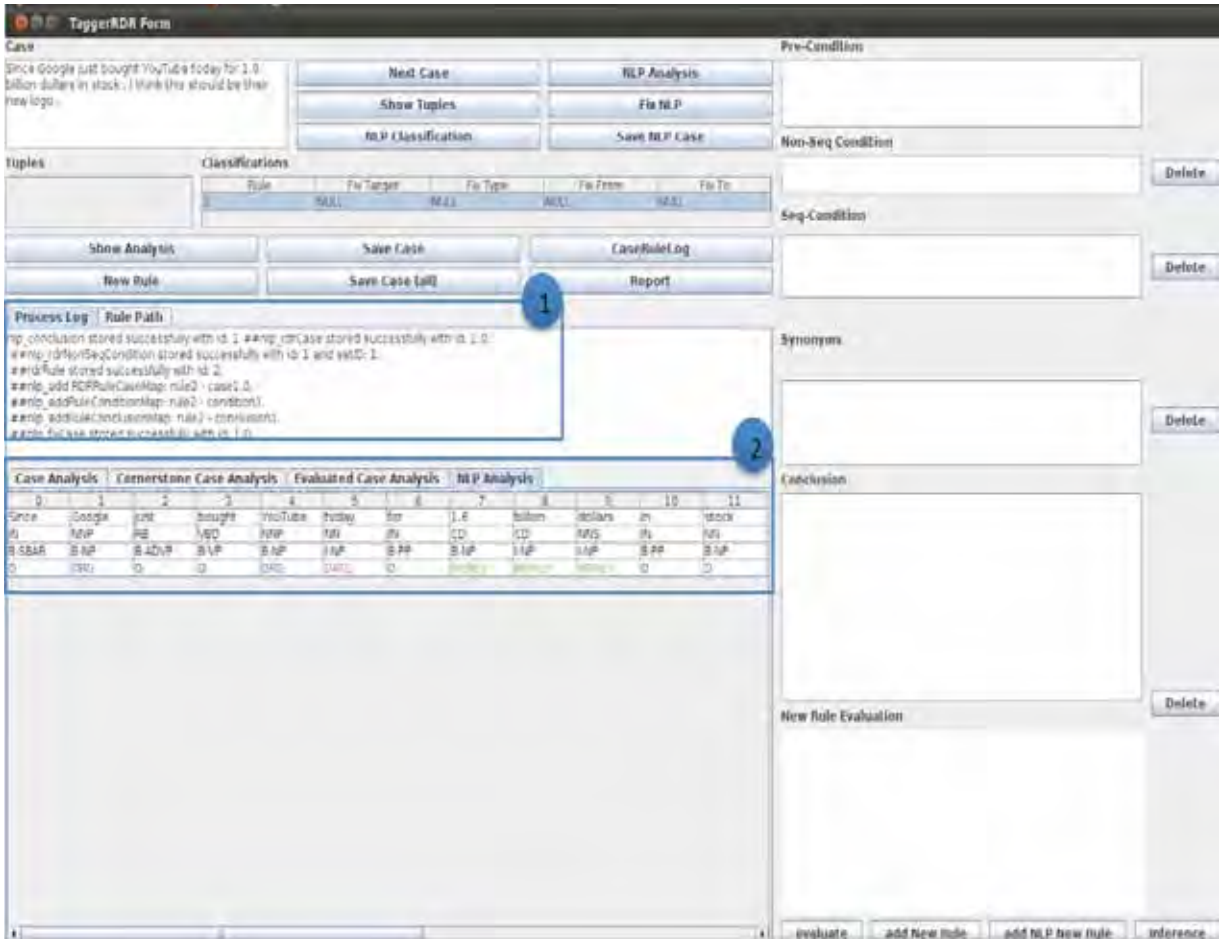

Figure 6.6. The KA process when the user confirms the effect of the new rule on the case

Step 1: The user can confirm where the new rule is saved in the KB using the process log.

Step 2: The user can click the ‘NLP Analysis’ button and view the ‘NLP Analysis’ table to check how the NE error is corrected after applying the RDR rule. ‘YouTube’ is correctly tagged as an ORGANISATION NE tag.

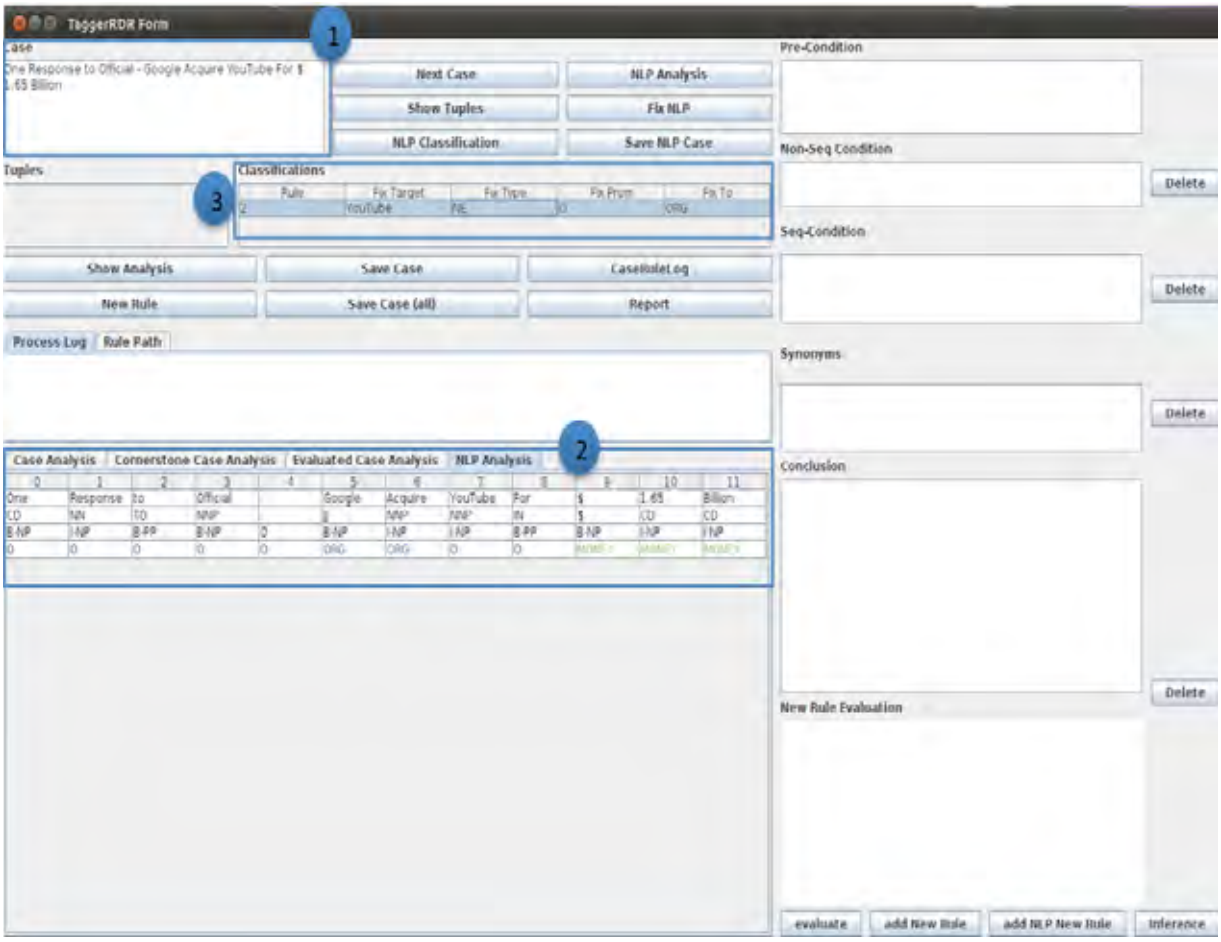

Figure 6.7. The KA process when new case entered into the system

Step 1: The given sentence '*One Response to Official – Google Acquire YouTube for $ 1.65 Billion*' is displayed.

Step 2: After clicking the 'NLP Analysis' button, the user views the 'NLP Analysis' table to check whether any NE errors occurred. The user notices that 'YouTube' is not tagged.

Step 3: As the user clicks the 'NLP Classification' button, the rule R2 fired with '('YouTube', NE, O, ORG)' conclusion from KB. The user agrees that the fired rule is correct.

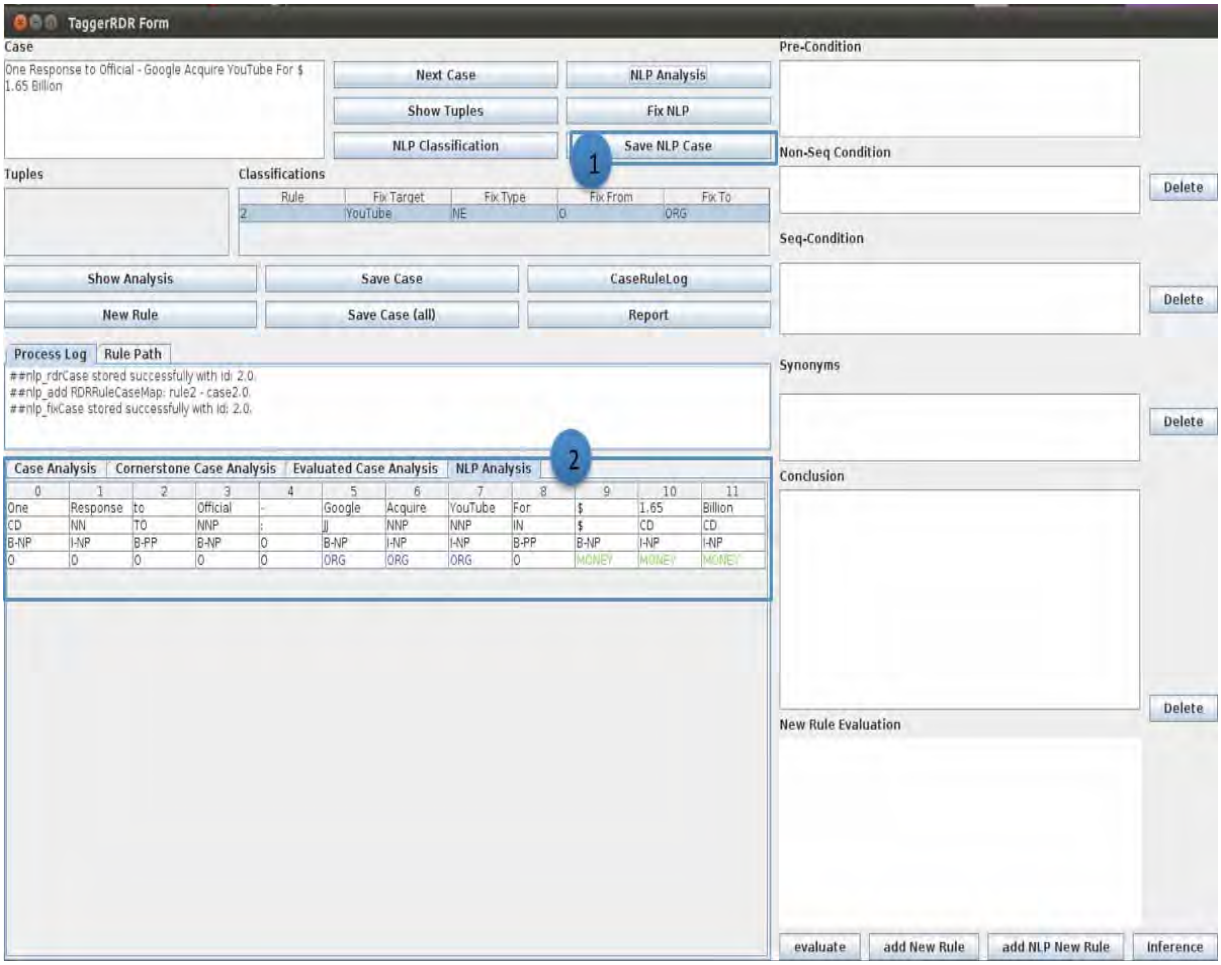

Figure 6.8. The KA process when the user saves the given case for the correctly fired rule

Step 1: As the fired rule R2's classification result is correct, the user saves the current case to R2.

Step 2: The user can click the 'NLP Analysis' button to refresh the 'NLP Analysis' table. As R2 is applied to the current case, the NE error is corrected. 'YouTube' is correctly tagged as an ORGANISATION NE tag. However, the case still has an NE boundary detection error as 'Google Acquire YouTube' is tagged as an ORGANISATION NE. The user decides to create an exception rule R3 under the currently fired rule R2 to correct the NE error further.

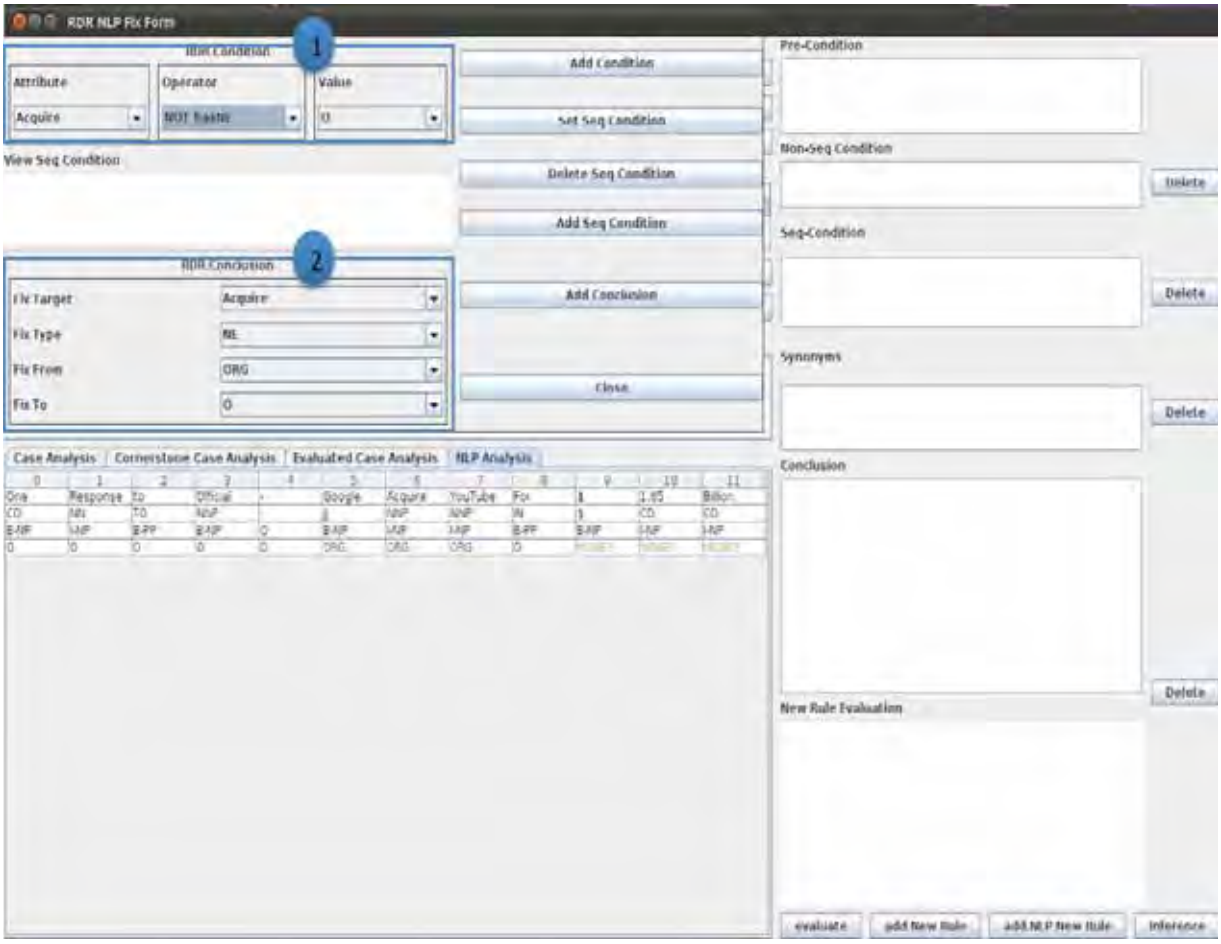

Figure 6.9. The KA process when the user clicks the 'fix NLP' button to create a new rule and then the 'RDR New Rule Form' window pops up

Step 1: The user sets the rule condition 'IF ('**Acquire**' NotHasNE '**O**')'.

Step 2: The user sets the rule conclusion 'THEN (Acquire, NE, ORG, O)'.

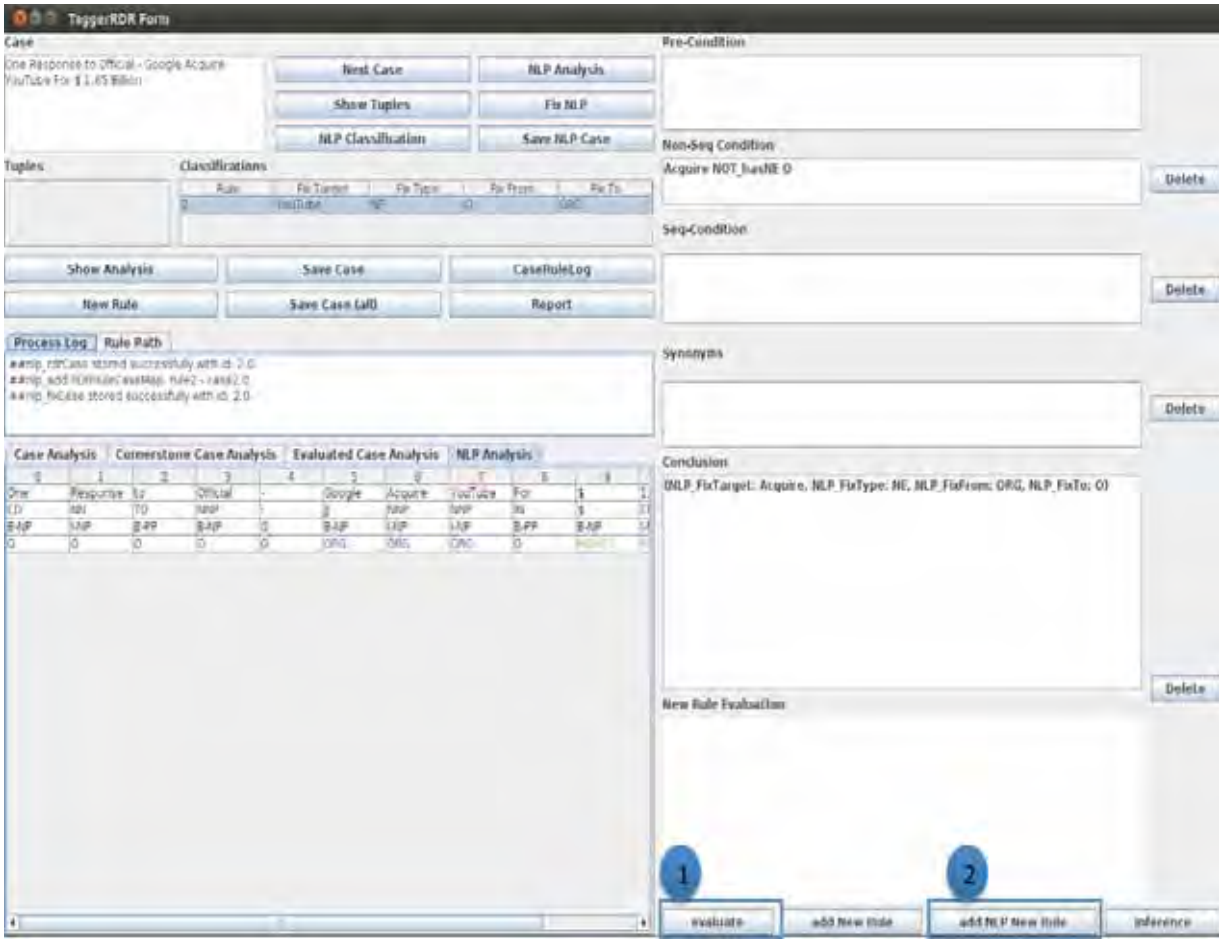

Figure 6.10. The KA process when evaluating the new rule created and saving it into the KB

Step 1: After setting an exception rule R3's condition and conclusion, the user clicks the 'evaluate' button to check whether the exception rule R3 under R2 conflicts with existing KB.

Step 2: As no conflict occurs, the user clicks the 'add NLP New Rule' button to save the exception rule R3 under the parent rule R2.

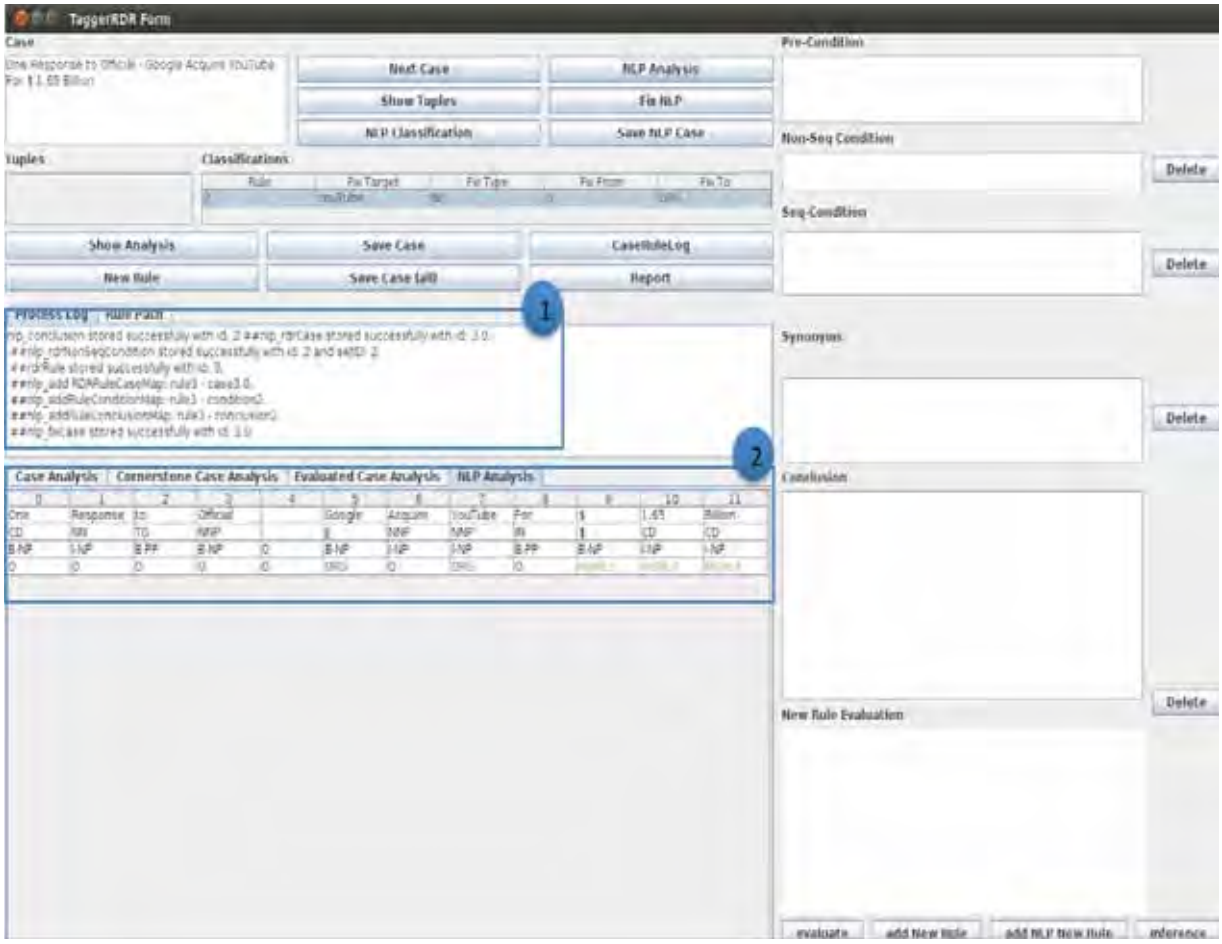

Figure 6.11. The KA process when the user confirms the effect of the new rule on the case

Step 1: The user can confirm where the exception rule R3 has been saved into the KB using the process log.

Step 2: The user can click the ‘NLP Analysis’ button and view the ‘NLP Analysis’ table to check how the NE error is corrected after applying the RDR rule. ‘Acquire’ is correctly tagged as an O NE tag from an ORGANISATION NE tag.

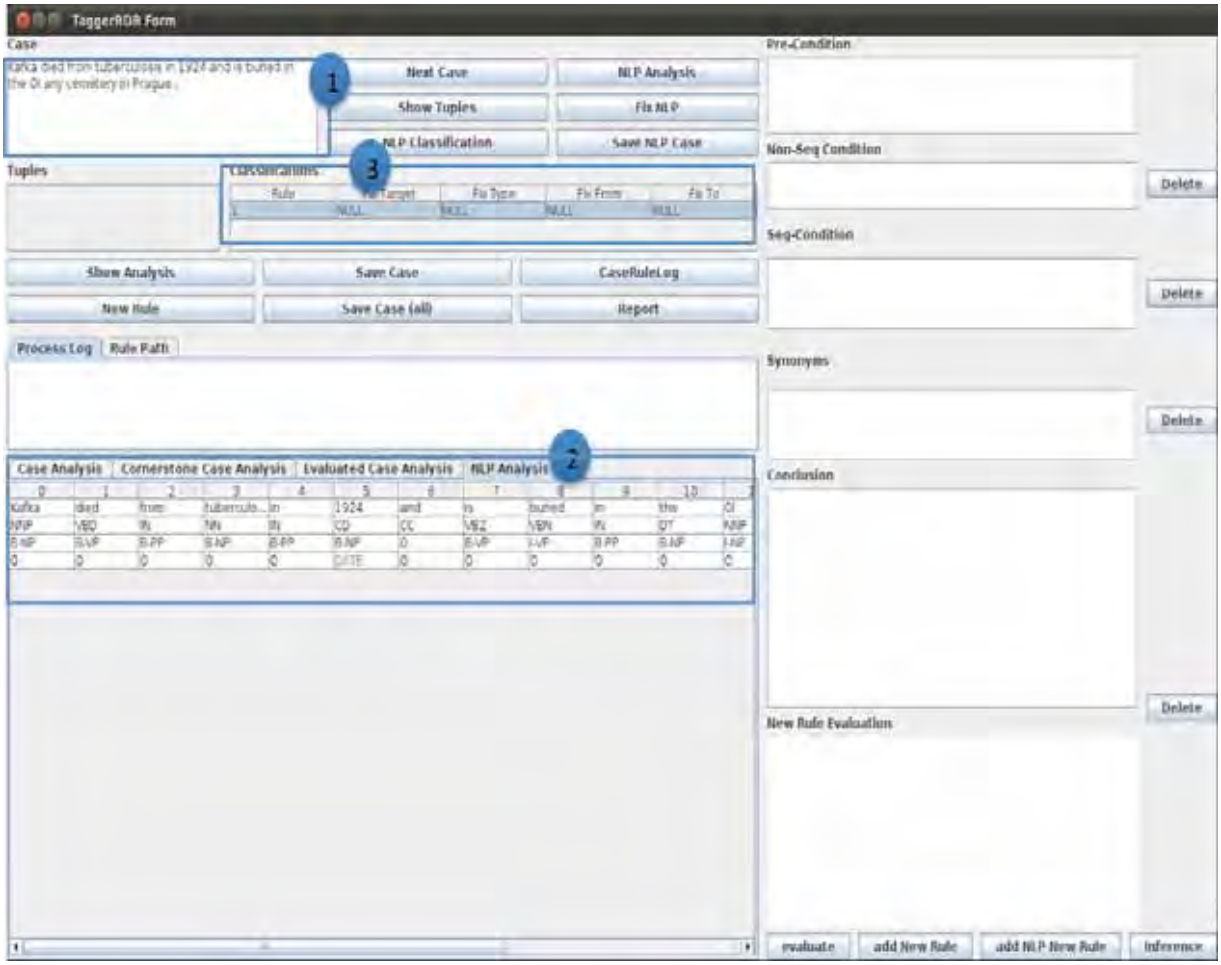

Figure 6.12. The KA process when a new case is entered into the system

Step 1: The given sentence '*Kafka died from tuberculosis in 1924 and is buried in the OI any cemetery in Praque.*' is displayed.

Step 2: After clicking the 'NLP Analysis' button, the user views the 'NLP Analysis' table to check whether any NE errors occurred. The user perceives that 'Kafka' is not correctly tagged as a PERSON NE.

Step 3: When the user clicks the 'NLP Classification' button, the rule R1 is fired with a NULL classification result. The user decides this is an incorrect classification. Because no rule is fired

to fix the NE error case, the user decides to create a new RDR rule R4 under the default rule R1 to correct it.

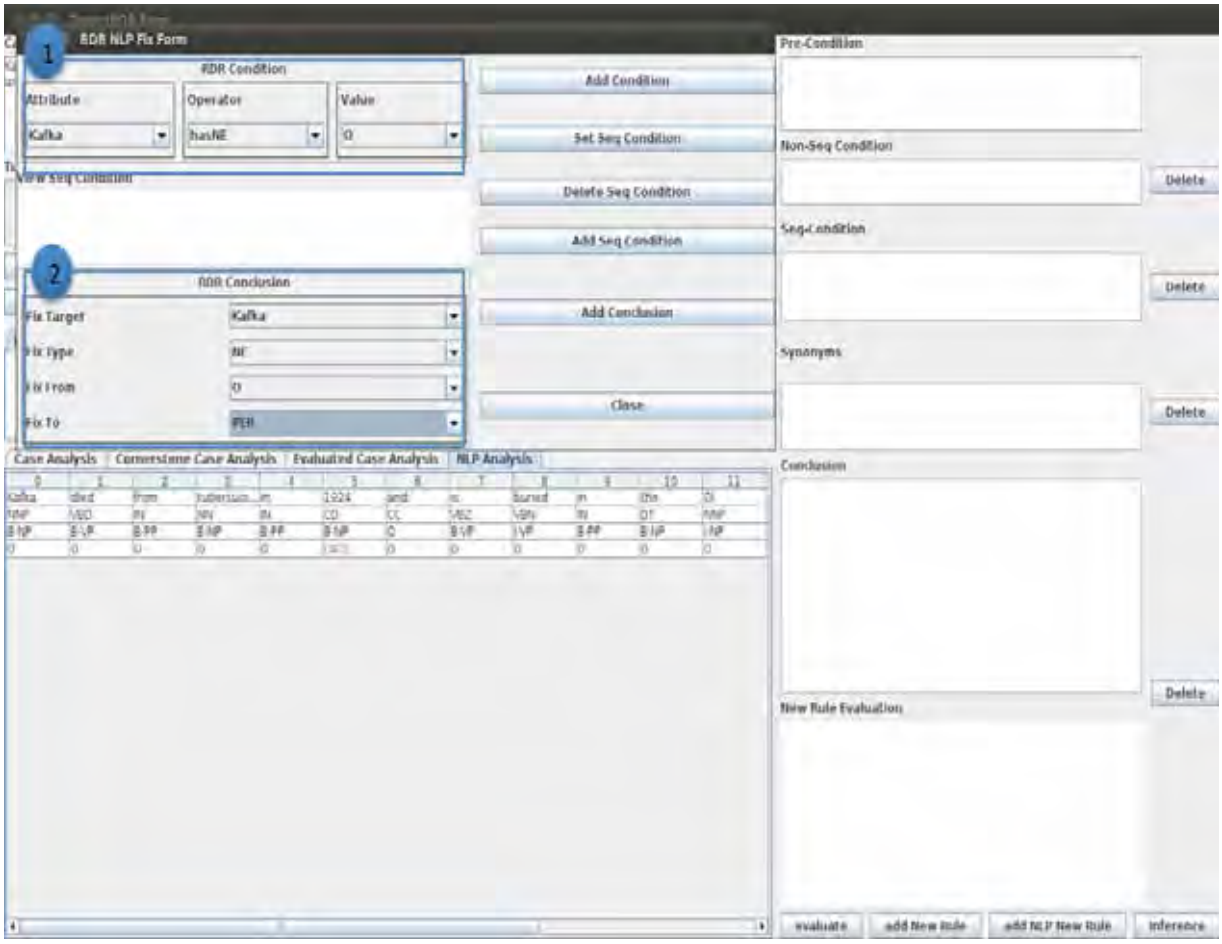

Figure 6.13. The KA process when the user clicks the ‘fix NLP’ button to create a new rule, then the ‘RDR New Rule Form’ window pops up

Step 1: The user sets the rule condition ‘IF (‘**Kafka**’ hasNE ‘**O**’)’.

Step 2: The user sets the rule conclusion ‘THEN (Kafka, NE, O, PERSON)’.

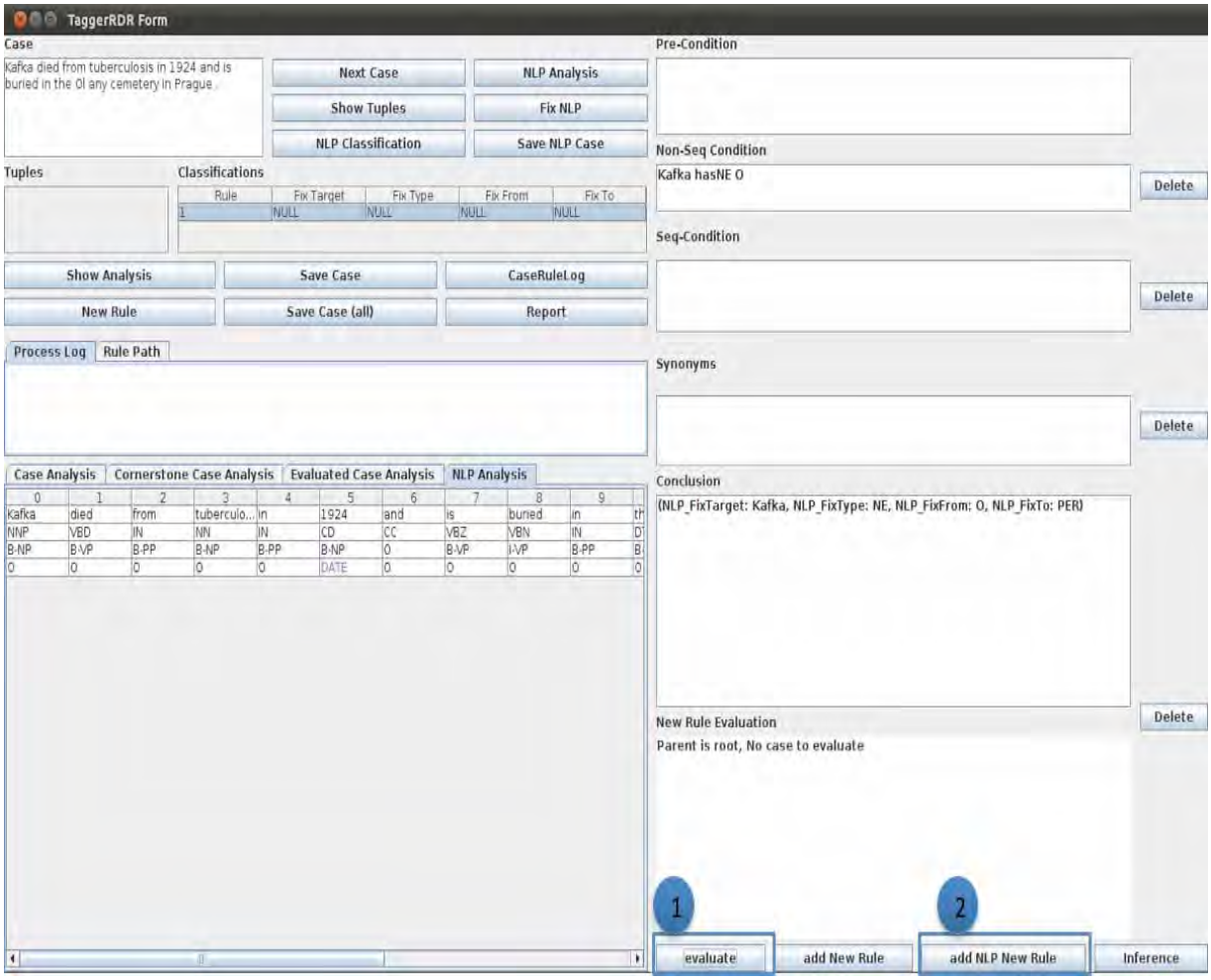

Figure 6.14. The KA process when evaluating the new rule created and saving it into the KB system

Step 1: After setting the new rule R4's condition and conclusion, the user clicks the 'evaluate' button to check whether the new rule R4 under the default rule R1 conflicts with existing KB.

Step 2: As no conflict occurs, the user clicks 'add NLP New Rule' button to save the new rule R4 under the default rule R1.

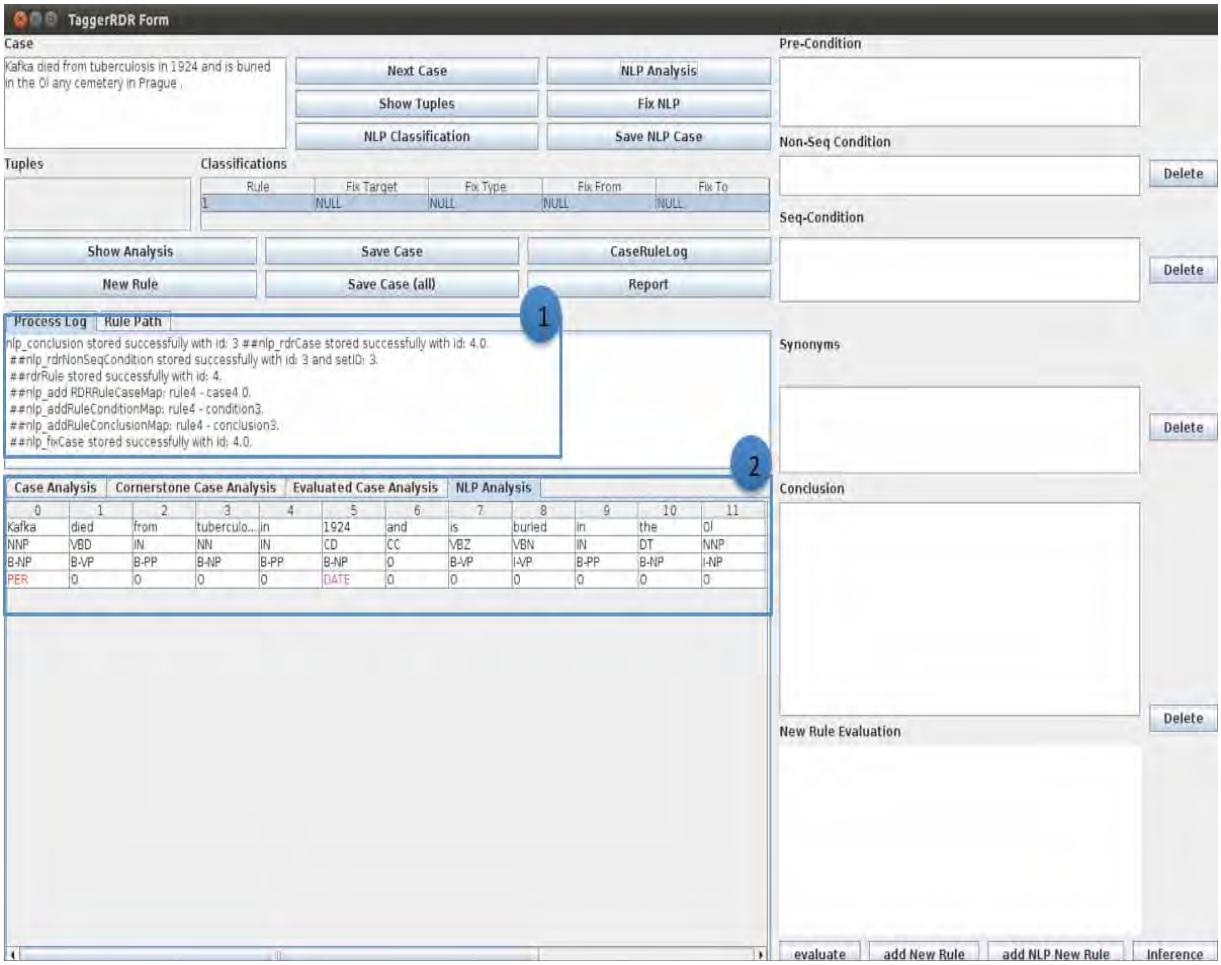

Figure 6.15. The KA process when the user confirms the effect of the new rule on the case

Step 1: The user can confirm where the exception rule R4 is saved into the KB by looking at the process log.

Step 2: The user can click the ‘NLP Analysis’ button and view the ‘NLP Analysis’ table to check how the NE error is corrected after applying RDR rule. ‘Kafka’ is correctly tagged as a PERSON NE tag from an O NE tag.

6.5 Performance Comparison of Three NER Systems

6.5.1 Dataset

Wikipedia documents are retrieved randomly from the 'U.S. President' category to be used as a test dataset. Non-sentential data such as HTML tags are removed to clean the dataset. The complete corpus contains a total of 87161 words, distributed in 1153 paragraphs. A gold standard for the test corpus is established by manual annotation following the Named Entity Recognition task definition (Chinchor et al., 1999). The corpus is annotated with three basic types of named entity classes: PERSON, ORGANISATION and LOCATION. Among a total of 8227 annotated named entities, 4147 named entities are annotated as Person, while 1968 and 2112 entities are annotated as Location and Organisation, respectively.

The reason that Wikipedia documents are chosen as an initial web data source for the experiment is because they are written with a less strict writing style than journalistic texts so the sentence structure is freer, sometimes colloquial, although it contains high quality and constantly updated contents. Thus we assume that if an NER system that is trained on fine journalistic documents performs well with Wikipedia documents, it might also perform relatively well with Web documents. Although Web documents are more heterogeneous and contain more incomplete sentences and various noises, Wikipedia documents are closer to Web documents than journalistic documents.

6.5.2 NER System Setting

We choose to compare the three best publicly available NER systems: Stanford NER system, LBJ NER system and GATE NER system.

The Stanford NER system is based on the use of Gibbs sampling for inference in a Conditional Random Field (CRF) machine learning technique (Finkel et al., 2005). In this experiment, we chose the classifier with 3 types of NE classes: PERSON, ORGANISATION and LOCATION trained on datasets from CONLL, MUC6, MUC7, and ACE. The selected classifier is the version that uses Alexander Clark's distributional similarity code to improve performance (Clark and Tim, 2003).

The LBJ NER system employs a number of expressive features based on a baseline learning system such as 'context aggregation', 'extended predication history', 'two-stage prediction aggregation', 'gazetteer match' and 'word class model' (Ratinov and Roth, 2009). The Open source version provides four types of pre-compiled models, which are trained on the CONLL dataset, and it classifies four types of NE classes: PERSON, ORGANISATION, LOCATION, and MISC. One of innovative feature of the LBJ NER system is that it uses 30 rich gazetteers. It provides a collection of 14 gazetteers, which are extracted from the Web and cover common entities: countries, monetary units, temporal expressions etc. These gazetteers have excellent accuracy, but are limited in coverage. To improve coverage, it also provides another 16 gazetteers, which are extracted, from the Wikipedia corpus. These Wikipedia-based gazetteers contain over 1.5M entities. In order to accelerate the performance with gazetteers, LBJ NER system developed a technique for injecting non-exact string matching to the gazetteers (Ratinov and Roth, 2009).

The GATE NER system (also called as 'ANNIE') is a rule-based entity extraction module integrated in the GATE framework. It consists of a set of default processing resources (i.e. tokeniser, sentence splitter, part-of-speech tagger, gazetteer, finite state transducer, semantic tagger, orthomatcher and coreferencer) that can be used individually or together to extract named

entities. These resource processes can be substituted by other plug-ins or even disabled. They are controlled by external resources such as grammars or rule sets (Bontcheva et al., 2004). The gazetteer consists of lists such as countries, cities, organisations, job titles, etc. It contains not only a list of entities, but also lists of useful indicators (trigger words) such as typical person titles (e.g. 'Mr.' and 'Dr.') and company designators (e.g. 'Ltd'). The semantic tagger annotates named entity instances by applying hand-crafted rules which are written in the JAPE (Java Annotations Pattern Engine) language. It describes a set of pattern/action rules using a regular expression-based-process to match and annotate named entities.

The Stanford Named Entity Recogniser version 1.1[11], LBJ Named Entity Tagger version 1.0[12] and GATE ANNIE version 5.0[13] were used for experiment. Among the various classifiers provided by each system, 'ner-eng-ie.erf-3-all2008.distsim.ser.gz' classifier is the version of Stanford NER system used and 'allLayer1.config' is chosen for LBJ NER system.

6.5.2 NER Performance Comparison

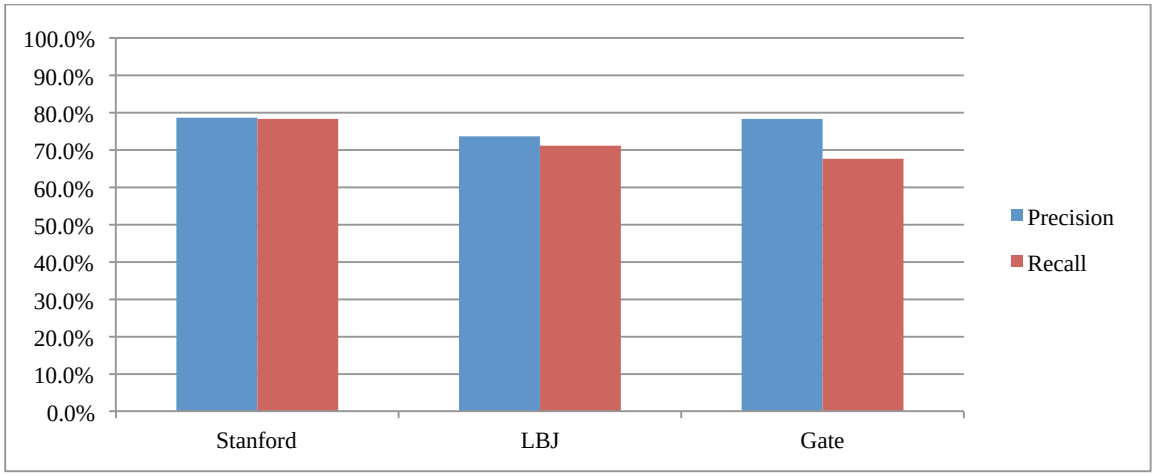

Figure 6.16. Precision and recall comparison of three NER systems

[11] http://nlp.stanford.edu/software/CRF-NER.shtml
[12] http://l2r.cs.uiuc.edu/~cogcomp/download.php?key=FLBJNE
[13] http://gate.ac.uk/download/index.html

Figure 6.16 illustrates the precision and recall comparison of the three NER systems. The Stanford NER system showed the best performance among the three systems and its precision and recall are balanced. The LBJ NER system achieved the lowest precision score among the three NER systems. It mainly produced entity boundary detection errors. The GATE NER system had the largest differences between precision and recall performance. While the GATE NER system recognises named entities accurately with correct type classification and text boundary detection, it often missed named entities, which should be annotated ('silent cases').

Data type	Data set	Stanford NER	LBJ NER	GATE NER
Journalistic documents	Reuters 2003 Test	87.04	90.74	-
	Reuters 2003 Dev	92.36	93.94	-
	News article	-	-	80~90
Web documents	Web pages	72.50	74.89	-
	Wikipedia	**78.57**	**72.40**	**72.65**

Table 6.3. F1 score comparison of three NER systems

Table 6.3 summarises the performance of the three NER systems on various sources of data. The performance comparison between the Stanford NER system and LBJ NER system on 'Reuters data set' and 'Web pages' is referenced from (Ratinov and Roth, 2009). The performance result of the GATE NER system on news articles is from (Cunningham et al, 2002). Since the GATE NER system is rule-based and usually needs to be extended for specific domains such as business, we could not obtain the original GATE NER system's performance evaluation for the standard test set.

The performance comparison between the Stanford NER system and the LBJ NER system on the Reuters dataset was evaluated only by ‘exact text boundary match’ excluding ‘exact type classification’ evaluation. On the other hand, the experiment on Web pages was evaluated only by ‘exact type classification’ excluding ‘exact text boundary match’ evaluation. In the experiment on the Wikipedia corpus, we used a more strict evaluation method by considering ‘exact type classification’ and ‘exact text boundary detection’ because both cause IE extraction errors. Note that due to different evaluation methodologies applied, the F1 score in the table 6.3 needs to be considered in the light of different strictness of evaluation.

The Stanford NER system achieved 87.04 and 92.36 F1 scores on Reuters and a 78.57 F1 score on the Wikipedia corpus. The performance difference between formal journalistic document and Wikipedia courpus is 8.47~13.79. Although the Wikipedia documents contain relatively high quality grammar and contents, the Stanford NER system had a performance drop of around 10%. Because the system is trained on old journalistic documents, it suffers from coverage limitation on the new vocabularies. The LBJ NER system had 90.74 and 93.94 F1 scores on Reuters and a 72.40 F1 score on the Wikipedia corpus. The performance difference between journalistic documents and the Wikipedia corpus is 18.34~21.54, even larger than the Stanford NER system. The GATE NER system achieved an 80~90 F1 score on news articles and 72.65 on the Wikipedia corpus. The performance difference between journalistic documents and the Wikipedia corpus is 7.35~17.35. The LBJ NER system showed the largest performance difference between the Web and non-Web documents.

6.5.3 Error Analysis

6.5.3.1 Inconsistent NE Type Classification

The Stanford NER system often produced inconsistent entity type classifications while entity text boundary detection was correctly performed. For example, when the 'PERSON' type named entity instance, 'Obama', appeared several times in the same document, the Stanford NER system classified it as a 'PERSON', an 'ORGANISATION' or a 'LOCATION' type. This is because 'Obama' is a rare person name and the Stanford NER system is trained from old journalistic documents, which have coverage limitations for recent vocabularies. It is also due to the lack of trigger words. When there is a trigger word like 'President', the NE instance 'Obama' was correctly classified as a 'PERSON'.

6.5.3.2 Lack of Sufficient Management on Large Size Web Gazetteers

The LBJ NER system produced mainly entity text boundary detection errors. For example, in the case of 'St. John's Catholic Elementary School', the system only detected 'St. John's' as a 'LOCATION' named entity. For the case of 'John F. Kennedy', it detected 'John F.' and 'Kennedy' as 'PERSON' entities separately. This seems to be due to a heavy dependence on large size Web gazetteers without sufficient management. The Web gazetteers are treated as a 'bag of words' with lack of disambiguation management. The LBJ NER system uses 'Non-exact string matching' for gazetteers to accelerate the performance with the very large gazetteers (Ratinov and Roth, 2009). Although the look-up method accelerates the speed of searching the gazetteers, it decreases accuracy of exact text boundary detection.

6.5.3.3 Insufficient Analysis Methods for Informal Web Documents

The GATE NER system often missed named entities, which should be annotated ('silent cases') while it generally recognised named entities accurately with correct type classification and text boundary detection. The GATE NER system is rule-based and some of rules are heavily reliant on context clues such as trigger words and capitalisation which can easily be found in journalistic documents but not necessarily in Web documents. Thus Web documents that skipped trigger words or used capitalisation inadequately cannot be covered sufficiently by the existing context analysis techniques.

6.5.4 Discussion

The experimental results show that the Stanford NER system demonstrates the most balanced performance on both precision and recall and it also shows broader coverage compared to other NER systems. As we can observe from the Stanford NER system, while the machine learning-based approach achieves high and balanced performance, it suffers from inconsistency.

The critical point of this approach is that it is hard to interpret the cause of a particular error and fix it specifically. What can be done to improve such a system is to increase the domain training set and re-train the whole system, and expect the system to then cover the particular error found without introducing a new specific error. This approach does not seem ideal to handle the Web's informality efficiently.

Rosenfeld and Feldman (2007) proposed a solution to this problem by applying a rule-based NE validator to the extracted named entities from machine learning-based NER. The rules consist of a set of simple regular expression patterns. The NE validator simply checks whether the extracted named entities from ML-based system match with the rules or the gazetteers. They

used a relatively small rule base (about 300 lines of patterns in Perl) that required several days of human knowledge engineering effort.

In general, while a rule-based approach can easily fix particular errors on specific domain by adding a precise rule to cover the specific domain case, it may have critical limitations on maintenance for scalable and consistent rule structures. When a new rule is created or an existing rule is refined, the overall system may need significant rule debugging.

As described earlier, Ripple-Down Rules (RDR) (Compton and Jansen, 1990) is an approach for rule-based building that can maintain rule sets efficiently, thus achieving scalable rule systems. In an RDR-based system, rules are added incrementally to the knowledge base and consistency of previously seen cases in the knowledge base is maintained automatically during incremental learning. RDR thus seems an appropriate approach for providing a wrapper of the Stanford NER system, which is machine learning-based.

6.5 Experimental Setup

The Web dataset used is derived from the MIL dataset developed by Bunescu and Mooney (2007). They collected 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. Details of the MIL dataset are described in section 4.4. In MIL dataset, each sentence has one pair of entities manually identified for their relation extraction task, but those entity tags were removed for our NER task experiment. That is, there are no pre-defined tags in our Web dataset. Among 4260 sentences from positive example folder of the MIL dataset, we randomly selected

200 sentences as a training dataset and 341 different sentences as a test dataset.

The MIL dataset has been used widely to evaluate Web Information Extraction (WIE) (Zhu et al., 2009; Banko and Etzioni, 2008). We believe that the MIL dataset represents a reasonable approximation to the type of data for which an NER system might be used in a specific application. We expect in practice that documents of likely interest would be retrieved because they contained keywords and an NER would then be applied to those documents as part of the information extraction process. The MIL dataset provides an approximation of this, while at the same time being an accepted evaluation dataset. It should be noted that the MIL sentence selection selects far more entities than just those directly relating to the a1 and a2 tokens. For example the a1 and a2 tokens used included 6 person names but the sentences in the test dataset contained 60 different person names and so on with the other named entity classes.

6.6 Knowledge Base (KB) Construction

In practice, the RDRNER system would be used case by case, but in the experiments here we first tagged the 200 sentences using the Stanford NER system. 231 NE errors were identified and used to develop RDR rules. In processing of the 231 cases and adding rules as required, 44 new rules were created for the cases which received a NULL classification result and 39 exception rules were created for the cases which received an incorrect classification result from earlier rules. In total, 83 rules were added within 4 hours. KB construction time covered from when a case is called up until a rule is accepted as complete and the time spent is logged automatically. In a normal RDR system all the cases go through the rules and the expert corrects errors but some of the errors will be due to rules the expert added previously and there is plenty of evidence

that this occurs. The cornerstone cases help to minimise this, but the expert would need to store all cases to reduce it to zero. In the RDRNER system, we pull out the initial error cases and write rules just to fix these, but ideally we should run all training cases whether correct according to the Stanford NER system or not, to make sure we had no errors.

6.7 RDRNER Performance

After the initial KB was built, the 341 sentences of the test dataset were run on the RDRNER system. The test dataset contains 857 NEs including 150 PERSON, 499 ORGANISATION, 112 LOCATION, 41 MONEY, 40 DATE, 9 PERCENT, 2 TIME.

	Person	Organisation	Location	Money	Date	Time	Percent	Total
P	95.17%	91.51%	86.36%	100.00%	89.74%	100%	100%	92.12%
R	92.00%	88.58%	84.82%	92.68%	87.50%	100%	88.89%	88.68%
F1	93.48%	90.48%	85.50%	96.37%	88.99%	100%	94.18%	90.48%

Table 6.4. The performance of the RDRNER system on seven named entity classes

Table 6.4 presents the performance of the RDRNER system on seven NE classes on the test dataset. Overall, the RDRNER system achieved a 90.48% F1 score, while the Stanford NER system achieved a 76.96% F1 score on the same dataset (see table 6.1). That is, the RDRNER system improved the F1 score by 13.52% compared to the Stanford NER system. The 90.48% F1 score of the RDRNER system is close to the 90.8% F1 score, achieved by the Stanford NER system on the formal CONLL corpus (Ratinov and Roth, 2009). On average, the RDRNER

system achieved both high precision and recall. The difference between precision and recall is only around 4%.

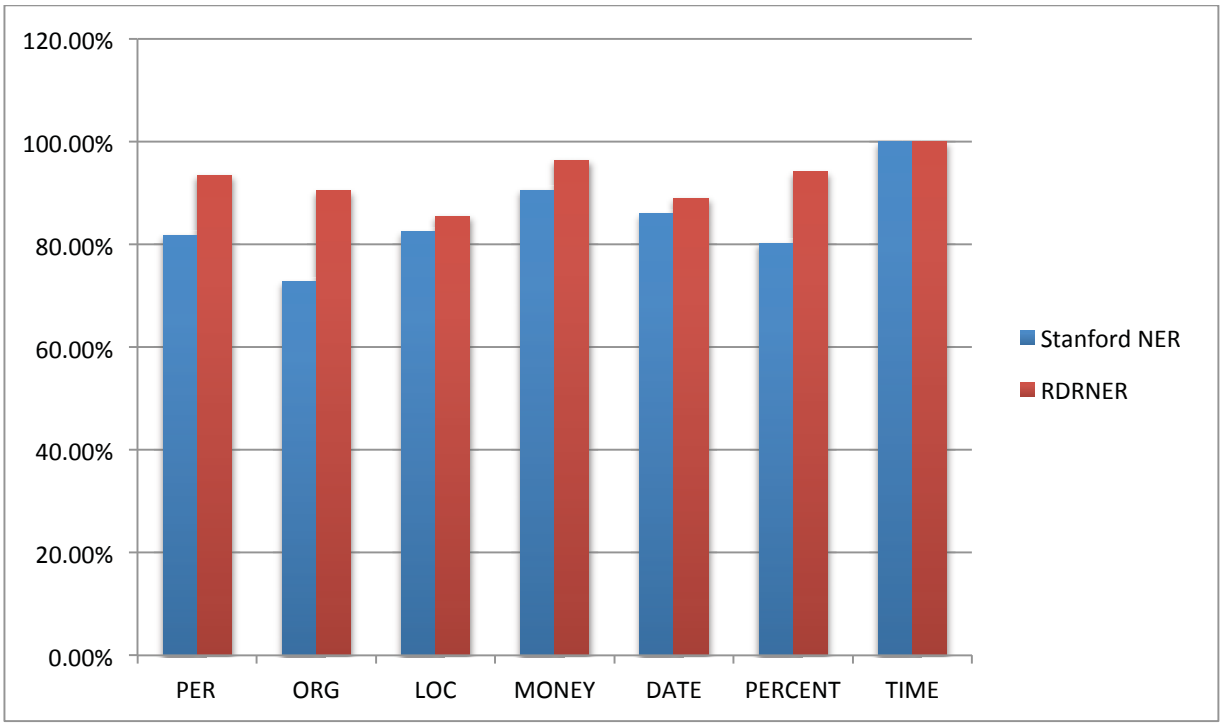

Figure 6.17. Performance improvement of the RDRNER system from the Stanford NER system

Figure 6.17 presents the performance improvement of the RDRNER system over the Stanford NER system on each of the seven NE classes. For all NE classes the RDRNER system improved the Stanford NER system performance except for 'TIME' where no rules were required as the Stanford NER system was already at 100%. ORGANISATION and PERSON NE classes had the biggest improvement.

6.8 How the RDRNER System Handles the Stanford NER System Errors

In this section, we show how the RDRNER system handles the five types of errors: New

vocabulary, ML inconsistency, Informal capital letter usage, Lack of trigger words and Web noise, which were discussed in section 6.1.

The following is an example of an error caused by new vocabulary. The word 'YouTube' was not tagged by the Stanford NER system because it was not contained in the training corpus and existing dictionary. A simple RDR rule is created to tag 'YouTube' as an ORGANISATION NE that function as extending the existing dictionary.

Error source	New vocabulary
Problem Case	'Google/**ORG** Buys/**O** YouTube/**O** for/**O** 1.65/**MONEY** Billion/**MONEY** Netscape.com/**O'**
RDR rule	IF ('**YouTube**' hasNE '**O**') THEN (YouTube, NE, O, ORG)
Resolved Case	'Google/**ORG** Buys/**O** YouTube/**ORG** for/**O** 1.65/**MONEY** Billion/**MONEY** Netscape.com/**O'**

The following is an instance of an error caused by ML inconsistency. In case 1, the word 'Kafka' was tagged correctly as PERSON NE but in case 2 it was tagged as LOCATION NE without clear reason by the Stanford NER system. A simple RDR rule is created to annotate 'Kafka' as a PERSON NE that amends the classification result derived from the ML-based Stanford NER system.

Error source	ML inconsistency
Problem Case	Case1: ‘(/**O** Encyclopedia/**ORG**)/**O** Kafka/**PER’** Case2: ‘(/**O** Almanac/**ORG** –/**O** People/**O**)/**O** Kafka/**LOC’**
RDR rule	IF (‘**Kafka**’ hasNE ‘**LOC**’) THEN (Kafka, NE, LOC, PER)
Resolved Case	‘(/**O** Almanac/**ORG** -/**O** People/**O**) /**O** Kafka/**PER’**

The following is an example of an error due to informal usage of capital letters. Because the word ‘Acquire’ started with a capital letter informally, it was treated as part of ORGANISATION NE and resulted in an NE boundary error by the Stanford NER system. A simple RDR rule is created to treat the word ‘Acquire’ as non NE which resolves the NE boundary error and correctly annotates two ORGANISATION NEs: ‘Google’ and ‘YouTube’.

Error source	Informal capital letter usage
Problem Case	‘Google/**ORG** Acquire/**ORG** YouTube/**ORG’**
RDR rule	IF (‘**Acquire**’ hasNE ‘**ORG**’) THEN (Acquire, NE, ORG, O)
Resolved Case	‘Google/**ORG** Acquire/**O** YouTube/**ORG’**

The following is an instance of an error caused by a lack of trigger words in informal Web documents. Because there are no trigger words available to tag ‘Prague wax museum’ as a LOCATION NE, only ‘Prague’ was tagged as a LOCATION NE by the Stanford NER system. Two simple example RDR rules are presented to extend the LOCATION NE boundary from ‘Prague’ to ‘museum’.

Error source	Lack of trigger words
Problem Case	‘Franz/**PER** Kafka/**PER** –/**O** Prague/**LOC** wax/**O** museum/**O’**
RDR rule	Rule1: (using ‘SEQ’ keyword) IF SEQ((‘**Prague**’ hasNE ‘**LOC**’) & (‘**wax**’ hasNE ‘**O**’) & (‘**museum**’ hasNE ‘**O**’)) THEN (wax, NE, O, LOC) and (museum, NE, O, LOC) Rule2: IF ((‘**Prague**’ hasNE ‘**LOC**’) & (‘**Prague**’ afterWD(+1) ‘**wax**’) & (‘**Prague**’ afterWD(+2) ‘**museum**’)) THEN (wax, NE, O, LOC) and (museum, NE, O, LOC)

Resolved Case	‘Franz/**PER** Kafka/**PER** –/**O** Prague/**LOC** wax/**LOC** museum/**LOC’**

The following shows three examples of errors caused by noise. Case 1 shows that ‘Googld’ not tagged due to a spelling error. Rule 1 is generated to tag the ‘Googld’ as an ORGANISATION NE when there is the word ‘Google’ in the same sentence is tagged as an ORGANISATION NE. Future conflict caused by this rule could be resolved by adding another exception rule under the rule as explained in section 6.3.3. Case 2 demonstrates that ‘Google’ was not tagged due to the symbol ‘+’. Two simple rules (Rule 2 and Rule 3) are created to amend it. Case 3 presents a NE boundary error due to an unknown abbreviation ‘Bn.’. Rule 4 is generated to fix the boundary error by adding the ‘Bn.’ as part of MONEY NE when there exist a $ sign within two tokens before ‘Bn.’.

Error source	Web noise (spelling error, various symbols and abbreviations)
Problem Case	Case 1: spelling error ‘This/**O** morning/**O** Googld/**O** held/**O** a/**O** webcast/**O** and/**O** conference/**O** call/**O** session/**O** with/**O** Eric/**PER** Schmidt/**PER** (/**O** Google/**ORG** CEO/**O**) /**O’** Case 2: symbols ‘Google/**O** +/**O** YouTube/**O** :/**O** Day/**O** Two/**O’**

	Case 3: abbreviations ‘Google/**ORG** bought/**O** YouTube/**ORG** for/**O** $/**MONEY** 1.6/**MONEY** Bn./**O** Sweet/**O’**

RDR rule	Rule 1 for case 1:
	
	IF ((‘**Google**’ hasNE ‘**ORG**’) &
	
	(‘**Googld**’ hasNE ‘**O**’))
	
	THEN (Googld, NE, O, ORG)
	
	Rule 2 for case 2:
	
	IF (‘**Google**’ hasNE ‘**O**’)
	
	THEN (Google, NE, O, ORG)
	
	Rule 3 for case 2:
	
	IF (‘**YouTube**’ hasNE ‘**O**’)
	
	THEN (YouTube, NE, O, ORG)
	
	Rule 4 for case 3:
	
	IF ((‘**Bn.**’ beforeWD(+2) ‘**$**’) &
	
	(‘**Bn.**’ hasNE ‘**O**’))
	
	THEN (Bn., NE, O, MONEY)

Resolved Case	Case 1: 'This/**O** morning/**O** Googld/**ORG** held/**O** a/**O** webcast/**O** and/**O** conference/**O** call/**O** session/**O** with/**O** Eric/**PER** Schmidt/**PER** (/**O** Google/**ORG** CEO/**O**)/**O'** Case 2: 'Google/**ORG** +/**O** YouTube/**ORG** :/**O** Day/**O** Two/**O'** Case 3: 'Google/**ORG** bought/**O** YouTube/**ORG** for/**O** $/**MONEY** 1.6/**MONEY** Bn./**MONEY** Sweet/**O'**

6.9 Discussion

Table 6.4 shows that the RDRNER system achieves high precision and recall after only 4 hours initial KB construction by a user working through a small training dataset. Given the very rapid training we propose that rather than simply having to accept an inadequate level of performance delivered by a general purpose NER tool, and the results of this flow on through the whole information extraction task; it is a worthwhile and viable alternative to very rapid add rules to specifically cover the domain of interests.

For NER, the use of lexical features such as dictionary and trigger words is one of main techniques. As discussed in section 2.2.2, recent studies have focused on creating gazetteers automatically using Web sources, which increases the size of gazetteers dramatically. With the current simple 'bag of words' approach, the increased size of gazetteers cannot be utilised efficiently because of problems with word sense disambiguation. Moreover, there is a lack of trigger words on the Web so it is difficult to utilise lexical features by pre-defining typical trigger

words on informal Web corpus.

The RDRNER system, however, can efficiently utilise lexical features for informal Web documents. Section 6.8 showed examples of the RDRNER system using lexical features in rule creation. One could perhaps characterise some of the rules we added as equivalent to adding words to a gazetteer; however, in the rule creation any combination of conditions can be used providing a much more sophisticated function than simply adding words to a gazetteer. Secondly, if the vocabulary causes an incorrect classification result with other cases seen later, we can simply add an exception rule to correct the classification, ending up with something much more sophisticated than a gazetteer. Similarly, when there is a lack of pre-defined trigger words in Web documents, the RDRNER system can be used to identify other trigger words, but again with a rule structure allowing a conjunction of conditions to be used.

The RDRNER system is designed to be trained on a specific domain of interest. We have not yet explored whether the rules for new domains should be added to rules for a previous domain, or whether it is better to start again (but again wrapping the rule system around a general tool like the Stanford NER system). As suggested in the previous chapter it would be interesting to see if it was possible to extend the domain coverage by using a version of crowd sourcing, whereby large numbers of people on the Web might contribute rules.

The RDRNER system required very little effort; rule creation is very simple and rapid. In the study here it took about three minutes on average to build a rule. Experience suggests that knowledge acquisition with RDR remains very rapid even for large knowledge bases (Compton et al., 2011). Rules can be updated as errors are uncovered, or when new vocabularies are created, or new meanings are attached to existing terminologies, or new colloquialisms come into use.

The alternative is to accumulate enough training data to retrain a system automatically, but this has its own demands. An error can be corrected manually the first time it occurs with an RDR system but with a machine learning-based system, one needs to keep monitoring cases until sufficient examples of each type of errors have been accumulated for the learning system. We suggest it is simpler to correct the error when it first occurs. The user is then monitoring a continuously improving system. When errors are sufficiently infrequent given domain requirements, monitoring can stop.

Chapter 7. Conclusion and Future Work

7.1 Conclusion

The Web contains a massive quantity of information in natural language and extracting valuable information from informally written Web text and storing it in machine-readable structured forms is one of the major research challenges. Web IE aims 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 corpora. Open IE (OIE) offers scalability for Web IE by developing relation-independent IE systems which extract all potential relations to reduce the amount of time necessary in finding desired information.

In this thesis, we have presented RDR's incremental learning technique applied for the OIE task to handle the Web's informality that has not been focused on in previous OIE systems. An RDR-based system requires no labelled training data and it is more likely to be able to handle

NLP tools errors and informally written Web documents as rules are generated by humans to deal with specific cases. In RDR rule creation, lexical features are mainly utilised, as it is difficult to handle the informality of Web text using NLP features such as part-of-speeches, noun/verb chunks and named entities. On the other hand this approach assumes a system is intended to be used in a specific area of interest where the appropriate rules can be written for the type of errors that occur in that domain (e.g. Departments of Health concerned with biosecurity monitor routinely monitor news reports to note any outbreaks of disease).

In this thesis, we have introduced three RDR-based systems: RDR-based Open Information Extraction (RDROIE) system, Hybrid RDR-based Open Information Extraction (Hybrid RDROIE) system, and RDR-based Named Entity Recognition (RDRNER) system. All three systems support incremental knowledge acquisition and efficient knowledge base maintenance.

The RDROIE system rapidly achieved and maintained very high precision after only 2.5 hours training by a user on a small dataset in the domain of interest and overall outperformed previous machine learning-based OIE systems using sophisticated automated training algorithms. The experimental results illustrated the advantage of an RDR approach. The RDROIE system can be trained on whatever text it is meant to classify and errors can keep on being corrected incrementally at minimum cost while the system is in use. In contrast, previous machine learning-based systems require very large amounts of training data and there may be no annotated training data available similar to the Web domain due to NLP tool errors on informal Web documents.

We have not evaluated the RDROIE system in an area where it has not been trained, and assume a particular RDROIE knowledge base would not perform well on a different domain.

Our second approach was to apply RDR's incremental learning technique on top of a general OIE system, so that the baseline performance would be the general system with the RDR correcting errors incrementally from the baseline system in a domain of interest. The system performs well where it has been trained, but when it comes across examples it has not been trained for, it falls back to the baseline of the general purpose system. In order to improve the state-of-the-art REVERB system, we developed the Hybrid RDROIE system where RDR used on top of the REVERB system. The Hybrid RDROIE system achieved high precision and recall after only 2 hours training on a small training dataset in a domain of interest. As the experimental results illustrated the improved performance after a very short training time, we suggest that rather than simply having to accept an insufficient level of performance delivered by the REVERB system in a specific domain, it is a worthwhile and practical alternative to very rapidly add rules to specifically cover the REVERB system's errors in that domain.

We then introduced the RDRNER system, which is a wrapper around the machine learning-based Stanford NER system as its performance degrades significantly with informal Web documents. The RDRNER system corrects the Stanford NER system again using RDR's incremental learning technique. The key advantages of this approach is that it can handle the freer writing style that occurs in Web documents and correct errors introduced by the Web's informal characteristics. In these studies the Ripple-Down Rules approach, with minimal low-cost rule addition improved the Stanford NER system's performance on informal Web document to the same level as its state-of-the-art performance on formal documents.

In summary in this thesis have demonstrated that an incremental learning approach with rules provided by humans can be used to efficiently handle errors that are difficult for more automated approaches. It also verified that the incremental approach used on top of a more automated

conventional system can improve the performance of the general systems in a specific application area. This leads to the general suggestion that if an incremental learning approach is used on top of the best current system, the performance of the overall system should be better than the state of the art general systems.

7.2 Future Work

To further improve the power of RDR-based OIE system, we have identified following directions for future work.

Generalised RDROIE Constructing an OIE system based on generalised RDR might solve both NLP correction and the OIE task with a single KB rather than handling those tasks with two or more KBs. In generalised RDR, a partial conclusion is added to a case, then the inference process rerun with the augmented case. With RDR the expert is likely to deal with individual errors, but generalised RDR's accumulative refinement through repeated inferencing cycles might support the system to construct a very high level of expertise.

Collaborative RDROIE One interesting approach might be to use a version of crowd sourcing, whereby large numbers of people on the Web might contribute rules. This might allow an RDR approach to be used much more broadly than in a specific domain of interest. However, no one has yet looked at how to develop RDR systems with multiple people providing knowledge. E.g. if person A adds a correction to a rule by person B, how do you know the change should be accepted? Is there evidence elsewhere in the knowledge base that rules by B are corrected less frequently by others than rules by A?.

RDROIE for noisier Web documents It would be an interesting challenge to see if the same approach can be used for knowledge bases for very noisy Web documents like Twitter. Twitter provides access to large volumes of data in real time, but is notoriously noisy, resulting in serious NLP tool errors. It often includes informal abbreviations, out-of-vocabulary words and ill-formed words which are difficult to identify using existing NLP tools. Applying the RDR technique should be helpful in correcting NLP tools errors and extracting real time information.

Bibliography

Agichtein, E. and Gravano, L. (2000). Snowball: Extracting Relations from Large Plain-text Collections. *In Proceddings of the 5th ACM International Conference on Digital Libraries.* pp. 85-94. ACM: San Antonio, Texas, USA.

Alfonseca, E. and Manandhar, S. (2002). An unsupervised method for general named entity recognition and automated concept discovery. *In Proceedings of the 1st International Conference on General WordNet*, Mysore, India.

Appelt, D. A., Hobbs, J. R., Bear, J., Israel, D. and Tyson, M. (1993). FASTUS: A finite-state processor for information extraction from real-world text. *In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, Washington DC.

Asahara, M. and Matsumoto, Y. (2003). Japanese Named Entity Extraction with Redundant Morphological Analysis. *In Proceedings of Human Language Technology conference - North American chapter of the Association for Computational Linguistics.*

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J. and Ives, Z. (2007). DBPedia: A Nucleus for a Web of Open Data. *In Proceedings of 6th International Semantic Web Conference*, pp. 11-15. Busan, Korea.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. *In the Proceedings of the 20th international joint conference on Artifical intelligence.*

Banko, M. and Etzioni, O. (2008). The Tradeoffs Between Open and Traditional Relation

Extraction. *In: ACL-08: HLT*.

Berant, J., Dagan, I. and Goldberger, J. (2011). Global learning of typed entailment rules. *In Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Portland, OR.

Bikel, D.M.,Schwartz R. and Weischedel, R.M. (1999). An Algorithm that Learns What's in a Name. *Machine Learning*, Vol. 1,3, pp. 211-231.

Bontcheva, K., Tablan, V., Maynard, D. and Cunningham, H. (2004). Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*. 10, pp. 349-373.

Borthwick, A. (1999). A Japanese named entity recognizer constructed by a non-speaker of Japanese. *In Proceedings of the IREX Workshop*, pp.187-193.

Bowden, P.R., Halstead, P., and Rose, T. (1996). Extracting conceptual knowledge from text using explicit relation markers. *In Proceedings of 9th European Knowledge Acquisition Workshop (EKAW-96)*, pp. 147-162. Nottingham, UK.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21, pp. 543-565.

Brin, S. (1998). Extracting patterns and relations from World Wide Web. *In Proceedings of SIGMOD Workshop on the World Wide Web and Databases*, pp. 172–183. Valencia, Spain.

Califf, M.E. and Mooney, R.J. (1997). Relational Learning of Pattern-Match Rules for Information Extraction. *In Proceeding of ACL-97 Workshop in Natural Language Learning*.

Cao, T.M. and Compton, P. (2005). A simulation framework for knowledge acquisition evaluation. *In 28th Australian Computer Science Conference (ACSC2005)*, pp. 353-360. Newcastle.

Chinchor, N., Brown, E., Ferro, L. and Robinson, P. (1999). Named Entity Recognition Task Definition, The MITRE Corporation and SAIC.

Cimiano, P., Handschuh, S. and Staab, S. (2004) Towards the self-annotating web. *In Proceedings of the 13th International Conference on World Wide Web*, pp. 462-471. ACM: New York, NY, USA.

Cimiano, P. Ladwig, G. and Staab, S. (2005). Gimme' the context: context-driven automatic semantic annotation with C-PANKOW, *In Proceedings of the 14th International Conference on World Wide Web*, pp. 332-341. ACM: Chiba, Japan.

Clark, A. and Tim, I. (2003). Combining Distributional and Morphological Information for Part of Speech Induction. *In Proceedings of the tenth Annual Meeting of the European Association for Computational Linguistics*.

Collot, M., and Belmore, N. (1996). Electronic Language: A New Variety of English. *In Computer-Mediated Communications: Linguistic, Social and Cross-Cultural Perspectives, Ed. S.C. Herring*, pp.129-46, Benjamins, Amsterdam.

Compton, P., Cao, T. and Kerr, J. (2004). Generalising incremental knowledge acquisition. *In Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B.H., Hoffmann, A., Yamaguchi, T and Yeap, W.K. (eds). University of Tasmania Eprints Repository, Auckland, pp. 44-53.

Compton, P., Kang, B., Preston, P. and Mulholland, M. (1993). Knowledge Acquisition without Analysis. Knowledg Acquisition for Knowledge-Based Systems, *In Proceedings of the 7th European Workshop, EKAW'93*, pp. 277-299.

Compton, P. and Jansen, R. (1988). Knowledge in context: a strategy for expert system maintenance. *In Proceedings of the 2nd Australian Joint Artificial Intelligence Conference, Lecture Notes in Computer Science 406*, pp. 292-306. Springer.

Compton, P., Edwards, G., Srinivasan, A., Malor, R., Preston, P., Kang, B., and Lazarus, L. (1992). Ripple Down Rules: turning Knowledge Acquisition into Knowledge Maintenance. *Artificial Intelligence in Medicine*, 4, pp. 47-59.

Compton, P. and Jansen, R. (1990). A philosophical basis for knowledge acquisition. *Knowledge Acquisition*, 2, pp. 241-257.

Compton, P., Kim, K.S. and Kang, B.H. (2011). Reconnecting domain knowledge and inference. (submitted)

Compton, P., Peters, L., Lavers, T. Kim, Y.S. (2011). Experience With Long-term Knowledge Acquisition, *In Proceedings of 6th International Conference on Knowledge Capture*, Banff, Alberta, Canada.

Compton, P. and Richards, D. (1999). Extending ripple down rules, *In 12th International Conference on Knowledge Engineering and Knowledge Managements*.

Compton, P. and Richards, D. (2000). Generalising ripple-down rules. *In: Knowledge Engineering and Knowledge Management: Methods, Models, Tools*, pp. 2-6.

Compton, P., Ramadan, Z., Preston, P., Le-Gia, T., Chellen, V. and Mulholland, M. (1998). A trade-off between domain knowledge and problem solving method power. *In 11th Banff Knowledge Acquisition Workshop (KAW) Proceedings*, Gaines, B. and Musen, M. (eds). Banff, Canada, pp. 1-19.

Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). GATE: An architecture for development of robust HLT applicatons. *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL),* pp. 168-175. Philadelphia, PA.

Cunningham, H., Maynard, D., Tablan, V., Ursu, C. and Bontcheva, K. (2001). Developing language processing components with gate (A user guide).

Dani, M. D., Faruquie, T. A., Garg, R., Kothari, G., Mohania, M. K., Prasad, K. H., Subramaniam, L. V. And Swamy, V. N. (2010). A Knolwedge Acquisition Method for Improving Data Quality in Services Engagements, *IEEE International Conference on Services Computing*, pp. 346-353.

Downey, D., Etzioni, O. and Soderland, S. (2005). A Probabilistic Model of Redundancy in Information Extraction. *In IJCAI*.

Edwards, G. (1996). Reflective Expert Systems in Clinical Pathology. MD Thesis, University of New South Wales.

Edwards, G. and Compton, P. (1993). PEIRS: a pathologist maintained expert system for the interpretation of chemical pathology reports. *Pathology*, 25, pp. 27-34.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D. and Yates, A. (2005). Unsupervised named-entity extraction from the Web: An experimental study. *Artif. Intell*. 165 (1), pp. 91-134.

Fader, A., Soderland, S. and Etzioni, O. (2011). Identifying Relations for Open Information Extraction. *In Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.

Finkel, J.R., Grenager T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. *In the Association for Computer Linguistics*.

Finlayson, A. and Compton, P. (2004). Incremental Knowledge Acquisition Using RDR for Soccer Simulation. *In Proceedings of Pacific Rim Knowledge Acquisition Workshop*. Auckland, New Zealand.

Frakes, W.B. and Yates, R.B. (1992). Information Retrieval: Data Structures and Algorithms. Prentice-Hall.

Gildea, D. (2001). Corpus variation and parser performance. *In Proceedings of the Conference on Empirical Methods in NLP (EMNLP)*.

Grishman, R and Sundheim, B. (1996). Message Understanding Conference-6: A Brief History. *COLING (International Conference on Computational Linguistics)*.

Hearst, M. (1992). Automatic Acquisition of Hyponyms from large Text Corpora. *In Proceedings of 14th International Conference on Computational Linguistics*, pp. 539-545. Nantes, France.

Hepple, M. (2000). Independence and Commitment: Assumptions for rapid training and execution of rule-based POS taggers. *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pp. 278-285.

Ho, V.H., Compton, P., Benatallah, B., Vayssiere, J., Menzel, L., and Vogler, H. (2009). An incremental knowledge acquisition method for improving duplicate invoices detection. *In Proceedings of the International Conference on Data Engineering*, pp. 1415–1418. Shanghai, China.

Ho, V., Wobcke, W. and Compton, P. (2003). EMMA: An E-Mail Management Assistant, *In IEEE/WIC International Conference on Intelligent Agent Technology*, Liu, J., Faltings, B., Zhong, N., Lu, R. and Nishida, T. (eds). IEEE, Los Alamitos, CA, pp. 67-74.

Hobbs, J., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., and Tyson, M. (1997). FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural Language Text. *In E. Roche and Y. Schabes (Eds.), Finite-state language processing*, pp. 383-406. Cambridge, Mass: MIT Press.

Hoffmann, A. and Pham, S.B. (2003). Towards topic-based summarization for interactive document viewing. *In Proceedings of the 2nd International Conference on Knowledge Capture*, pp. 28-35, New York, NY.

Hoffmann, R., Zhang, C and Weld, D. (2010). Learning 5000 relational extractors. *In ACL*.

Jiang, J. and Zhai, C. (2007). A systematic exploration of the feature spaces for relation extraction. *In Proceedings of the Conference on Human Language Technologies / North American Chapter of the Association for Computational Linguistics (HLT-NAACL).*

Kang, B. (1996) Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules. PhD Thesis, School of Computer Science and Engineering, University of New South Wales, Australia.

Kang, B., and Compton, P. (1994). A maintenance approach to case based reasoning. *In Advances in Case-Based Reasoning, 2nd European Workshop, EWCBR-94*, Chantilly, France, November 7-10, 1994. Lecture Notes in Computer Science 984, pp. 226-239. Springer.

Kang, B., Yoshida, K., Motoda, H. and Compton, P. (1997). A help desk system with intelligence interface. *Applied Artificial Intelligence*. 11, pp. 611-631.

Kang, B., Compton, P. and Preston, P. (1995). Multiple classification ripple down rules: evaluation and possibilities. *In Proceedings of the 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop*. Banff, Feb 26-March 3, 1, 17.1 – 17.20.

Kang, B., Compton, P and Preston, P. (1998). Simulated Expert Evaluation of Multiple Classification Ripple Down Rules. *In Proceedings of the Banff Knowledge Acquisition workshop on Knowledge Acquisition for Knowledge-Based* Systems, pp. EVAL 4, 1-19.

Kaplan, R. and Bresnan, J. (1982). Lexical-functional grammar: A formal system for grammatical representation. *In J. Bresnan (Ed.), The mental representation of grammatical relations*. Cambridge, MA: MIT Press.

Kazama, J.i. and Torisawa, K. (2007). ExploitingWikipedia as External Knowledge for Named Entity Recognition. *In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Prague, Czech Republic.

Kerr, J. and Compton, P. (2002). Interactive Learning when Human and Machine Utilise Different Feature Spaces. *In Proceedings of Pacific Rim Knowledge Acquisition Workshop*, Tokyo, Japan.

Kim, M. and Compton, P. (2004). Evolutionary document management and retrieval for specialized domains on the web. *International Journal of Human Computer Studies* 60(2), pp. 201-241.

Kim, M., Compton, P. and Kang, B. (1999). Incremental development of a web-based help desk system. *The 4th Australian Knowledge Acquisition Workshop*, pp 157-171. University of New South Wales, Sydney.

Kim, Y.S. and Kang, B.H. (2008). Search Query Generation with MCRDR Document Classification Knowledge, In EKAW, pp. 292-301.

Klein, D. and Manning, C.D. (2003). Accurate unlexicalised parsing. *In Proceedings of the ACL*.

Krzywicki, A. and Wobcke, W. (2009). Incremental E-Mail Classification and Rule Suggestion Using Simple Term Statistics, *In Proceedings of the 22nd Australasian Joint Conference on Advances in Artificial Intelligence*.

Kwok, R.B.H. (2002). Using Ripple Down Rules for Actions and Planning. *In Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence*, p. 604.

Lin, D. (1998). Dependency-based evaluation of Minipar. *In Workshop on the Evaluation of Parsing Systems*.

Lin, T., Mausam and Etzioni, O. (2010). Identifying Functional Relations in Web Text. *In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1266-1276, Cambridge, MA, October. Association for Computational Linguistics.

Liu, X., Zhang, S., Wei, F., Zhou, M. (2011). Recognizing Named Entities in Tweets. *In: 49th Association for Computational Linguistics*. pp. 359-367.

Marrero, M., Sánchez-Cuadrado , S. Lara, J.M. and Andreadakis, G. (2009). Evaluation of Named Entity Extraction Systems. *In Advances in Computational Linguistics: Research in Computing Science*.

Mikheev, A., Moens, M. and Grover, C. (1999). Named Entity recognition without gazetteers, *In Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics. Association for Computational Linguistics*, pp. 1-8. Bergen, Norway.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*. 30, pp. 3-26.

Nadeau, D., Turney, P.D. and Matwin, S. (2006). Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. *In Canadian Conference on Artificial Intelligence*.

Nguyen, D.P.T., Matsuo, Y. and Ishizuka, M. (2007). Relation extraction from wikipedia using subtree mining, *In: 22nd national conference on Artificial intelligence*, vol 2, pp. 1414-1420. AAAI Press.

Nguyen, D. Q., Nguyen, D. Q., Pham, S. B. And Pham, D. D. (2011). Ripple Down Rules for Part-of-Speech Tagging. *In the Proceedings of the 12th international conference on*

Computational linguistics and intelligent text processing, pp. 190-201.

Nobata, C. and Sekine, S. (1999). Towards automatic acquisition of patterns for information extraction. *In International Conference of Computer Processing of Oriental Language.*

Park, S.S., Kim, Y.S. and Kang, B.H. (2004a). Web Document Classification: Managing Context Change, *In ICWI*, pp. 143-151.

Park, S.S., Kim, Y.S. and Kang, B.H. (2004b). Personalized Web Document Classification using MCRDR, *In Proceedings of the Pacific Knowledge Acquisition Workshop*, Kang, B.H., Hoffmann, A., Yamaguchi, T. and Yeap, W.K. (eds). University of Tasmania Eprints Repository, Auckland, pp. 63-73.

Pham, S.B. and Hoffmann, A. (2002). Classifying Scientific Citations by acquiring knowledge on natural language. *In Proceedings of Pacific Rim Knowledge Acquisition Workshop*. Tokyo, Japan.

Pham, S.B. and Hoffmann, A. (2004a). Extracting positive attributions from scientific papers. *In Proceedings of the Discovery Science Conference*. Springer-Verlag, pp. 169-182.

Pham, S.B. and Hoffmann, A. (2004b). KAFTIE: a new KA framework for building sophisticated information extraction systems. *In Engineering Knowledge in the Age of the Semantic Web. 14th International Conference, EKAW'04*. Springer-Verlag.

Pham, S.B. and Hoffmann, A. (2005). Incremental Knowledge Acquisition for Extracting Temporal Relations. *In Proceedings of Natural Language Processing and Knowledge Engineering*, pp. 354-359.

Pham, S.B. and Hoffmann, A. (2006). Efficient Knowledge Acquisition for Extracting Temporal Relations. *In Proceedings of 17th European Conference on Artificial Intelligence*, pp. 521-525. Riva del Garda, Italy.

Pollard, C. and Sag, I. (1994). Head-driven Phrase Structure Grammar. *Chicago, IL: University of Chicago Press*.

Ponzetto, S.P. and Strube, M. (2007). Deriving a Large Scale Taxonomy from Wikipedia. *In Proceedings of the 22nd national conference on Artificial Intelligence*. Vancouver, British Columbia, Canada.

Prasad, K. H., Faruquie, T. A., Joshi, S., Chaturvedi, S., Subramaniam, V., Mohania, M. (2011). Data Cleansing Techniques for Large Enterprise Datasets, *In Proceedings of the 2011 Annual SRII Global Conference*.

Ramadan, Z., Mulholland, M., Hibbert, D.B., Preston, P., Compton, P. and Haddad, P.R. (1998). Towards an expert system in ion-exculsion chromatography by means of multiple classification ripple-down rules. *Journal of Chromatography*, 804(1), pp. 29-35.

Rau, L.F. (1991). Extracting Company Names from Text. *In Proceedings of 6th IEEE Conference on Artificial Intelligence Applications*, IEEE Computer Society Press, Miami Beach, Florida.

Ratinov, L. and Roth, D. (2009). Design Challenges and Misconceptions in Named Entity Recognition. *In Proceedings of CONLL09*.

Ratnaparkhi, A. (1998). Maximum Entropy Models for Natural Language Ambiguity Resolution, PhD thesis, University of Pennsylvania.

Ratnaparkhi, A. (1999). Learning to parse natural language with maximum entropy models. *Machine Learning*, 34, pp. 151-175.

Ravin, Y. and Wacholder, N. (1996). Extracting Names from Natural-Language Text, *IBM Research Report 20338*.

Richards, D. (2009). Two decades of Ripple Down Rules research. *The Knowledge Engineering Review*, 2, pp. 159-184.

Richards, D and Compton, P. (1999). Revisiting Sisyphus I – an incremental approach to resource allocations using ripple-down rules. *In 12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Gaines, B., Kremer, R and Musen, M. (eds). SRDG Publications, University of Calgary, 7-7.1 – 7-7.20.

Richardson, M. and Domingos, P. (2006). Markov Logic Networks. *Machine Learning*. 62: pp. 1-2.

Riloff, E. and Schmelzenbach, M. (1998). An empirical approach to conceptual case frame acquisition. *In Proceedings of the 6th Workshop on Very Large Corpora*.

Ritter, A., Mausam and Etzioni, O. (2010). A Latent Dirichlet Allocation Method for Selectional Preferences. *In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.*

Rosenfeld, B. and Feldman, R. (2007). Using Corpus Statistics on Entities to Improve Semi-supervised Relation Extraction from the Web. *In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic.

Rozenfeld, B. and Feldman, R. (2008). Self-supervised relation extraction from the Web. *Knowledge and Information Systems*. vol 17 (1), pp. 17-33.

Schoenmackers, S., Etzioni, O., Weld, D.S. and Davis, J. (2010). Learning first-order horn clauses from web text. *In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1088-1098, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sekine, S. (2006). On-demand information extraction. *In Proceedings of the COLING/ACL on Main conference poster sessions*, pp. 731-738. Morristown, NJ, USA. Association for Computational Linguistics.

Shinyama, Y. and Sekine, S. (2006). Preemptive Information Extraction using Unrestricted Relation Discovery. *In Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 304-311. New York City, USA, Association for Computational Linguistics.

Shiraz, G.M. and Sammut, C. (1997). Combining Knowledge Acquisition and Machine Learning to Control Dynamic Systems. *In Proceedings of the 15th International Joint Conference on Artificial Intelligence, IJCAI'97*, pp. 908-913.

Shiraz, G.M. (1998). Building Controller for Dynamic Systems. Doctoral dissertation, University of New South Wales, Australia.

Snow, R., Jurafsky, D. and Ng, A.Y. (2005). Learning Syntactic Patterns for Automatic Hypernym Discovery. In Saul, L.K., Weiss, Y. and Bottou, L. (Eds.), Advances in neural information processing systems, 17. Cambridge, MA: MIT Press.

Soderland, S., Roof, B., Qin, B. Xu, S., Mausam and Etzioni, O. (2010). Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3), pp. 93-102.

Suchanek, F.M., Kasneci, G. and Weilum, G. (2007). YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. *In Proceedings of the 16th International Conference on World Wide Web*. Banff, Alberta, Canada.

Suchanek, F.M., Kasneci, G. and Weilum, G. (2008). YAGO: A Large Ontology from Wikipedia and WordNet. *Web Semant*. 6(3), pp. 203-217.

Suchanek, F.M., Sozio, M. and Weikum, G. (2009). SOFIE: a self-organizing framework for information extraction. *In Proceedings of the 18th International Conference on World Wide Web*. pp. 631-640.

Toral, A. and Mu˜noz, R. (2006). A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia. *In 11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. *In Proceedings of the North American Association for Computational Linguistics (NAACL)*, pp. 173–180.

Turney, P.D. (2001). Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. *In Proceedings of 12th European Conference on Machine Learning (ECML)*, pp. 491-502, Freiburg, Germany.

Wang, J.C., Boland, M., Graco, W. and He, H. (1996). Use of ripple-down rules for classifying medical general practitioner practice profiles repetition. *In Proceedings of Pacific Knowledge Acquisition Workshop PKAW'96*, Compton, P., Mizoguchi, R., Motoda, H. and Menzies, T. (eds). October 23-25, Coogee, Australia, pp. 333-345.

Wang, R.C. and Cohen, W.W. (2007). Language-Independent Set Expansion of Named Entities using the Web. *In IEEE International Conference on Data Mining*.

Wang, G., Yu, Y. and Zhu, H. (2007a). PORE: Positive-Only Relation Extraction from Wikipedia Text. *In Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference*. Busan, Korea.

Wang, G., Zhang, H., Wang, H. and Yu, Y. (2007b). Enhancing Relation Extraction by Eliciting Selectional Constraint Features from Wikipedia. *In Natural Language Processing and Information Systems*, pp. 329-340. Springer Berlin: Heidelberg.

Witten, H. I. (2011). Wikipedia and how to use it for semantic document representation. ASE 2011:1.

Wobcke, W., Krzywicki, A. and Chan, Y.W. (2008). A Large-Scale Evaluation of an E-Mail Management Assistant, *In Proceedings of International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 438-442.

Wu, F. and Weld, D.S. (2007). Autonomously Semantifying Wikipedia. *In Proceedings of the 16th ACM Conference on Information and Knowledge management*. Lisbon, Portugal.

Wu, F. and Weld, D.S. (2008). Automatically Refining the Wikipedia Infobox Ontology. *In Proceedings of the 17th International Conference on World Wide Web*. Beijing, China ACM.

Wu, F. and Weld, D.S. (2010). Open Information Extraction using Wikipedia. *In Proceedings of the 48th Annual Meetings of the Association for Computational Linguistics*, Uppsala, Sweden. pp. 118-127.

Xu, H. and Hoffmann, A. (2010). RDRCE: Combining Machine Learning and Knowledge Acquisition, *In: Pacific Rim Knowledge Acquisition Workshop*.

Zaanen, M.Y. and Moll´a, D. (2007). A Named Entity Recogniser for Question Answering. *In Proceedings of 10th Conference on Pacific Association for Computational Linguistics*.

Zacharias, V. (2008). Development and Verification of Rule Based Systems—A Survey of Developers. *RuleML 2008*, Orlando, Springer-Verlag, pp. 6-16.

Zhu, J., Nie, Z., Liu, X., Zhang, B., and Wen, J.-R. (2009). StatSnowball: a statistical approach to extracting entity relationships. *In the Proceedings of the 18th international conference on World Wide Web*, pp. 101-110. ACM: Madrid, Spain.

Appendix A

Taxonomy of Binary Relationships in Sent500 dataset (Banko and Etzioni, 2008)

Nearly 95% of 500 binary extractions were described using one of eight lexico-syntactic patterns.

NP refers to noun phrases, E*i* refers to entities, and Prep indicates a preposition.

Relative Frequency	Category	Simplified Lexico-Syntactic Pattern	Example
37.8	Verb	E1 Verb E2	X created Y
22.8	Noun+Prep	E1 NP Prep E2	X is birthplace of Y
16.0	Verb+Prep	E1 Verb Prep E2	X moved to Y
9.4	Infinitive	E1 to Verb E2	X plans to acquire Y
5.2	Modifier	E1 Verb E2 Noun	X is Y winner
1.8	Coordinate*n*	E1 (and|,|-|:) E2 NP	X-Y deal
1.0	Coordinate*v*	E1 (and|,) E2 Verb	X , Y merge
0.8	Appositive	E1 NP (:|,)? E2	X hometown : Y

Appendix B

Named Entity Recognition (NER) Evaluation Methodology

Precision measures the percentage of how many named entity found correctly by the system and it is defined as:

$$Precision = \frac{the\ number\ of\ correct\ guesses\ by\ system}{the\ number\ of\ guesses\ by\ system} * 100$$

Recall measures the percentage of how many of the total amount of named entities to be recognised was actually correctly guessed by the system and it is defined as:

$$Recall = \frac{the\ number\ of\ correct\ guesses\ by\ system}{the\ number\ of\ named\ entities\ in\ solution} * 100$$

F-score is a harmonic mean of precision and recall calculated over all entity types and it is defined as:

$$F - score = 2 * \frac{precision * recall}{precision + recall} * 100$$

There are three major NER performance evaluation methodologies used for MUC, IREX, CONLL and ACE conferences. Firstly, MUC scores two types of task separately: one task is to classify the correct named entity type (TYPE) and the other task is to detect exact boundary of named entity text (TEXT). For example, there are 5 named entities to be extracted in the given corpus and the system extracted 4 named entities. Among those 5 guessed entities, 2 are only

correctly classified (2 TYPE) and 1 is correctly classified and detected exact boundary (1 TYPE + 1 TEXT). In this case, evaluation will be calculated as:

$$Precision = \frac{3\,TYPE + 1\,TEXT}{4\,TYPE + 4\,TEXT} * 100 = 50\%$$

$$Recall = \frac{3\,TYPE + 1\,TEXT}{5\,TYPE + 5\,TEXT} * 100 = 40\%$$

$$F - score = 2 * \frac{0.5 * 0.4}{0.5 + 0.4} * 100 = 44.4\%$$

The MUC measure gives partial credit for the correct prediction of either named entity type classification or named entity boundary detection. On the other hand, IREX and CONLL use 'exact-match' scoring methodology. That is, it only counts score for the named entity prediction which satisfies both correct named entity type and named entity text boundary. For the previous example, there are 5 entities to be extracted, 4 entities are guesses by system and only 1 guess is correctly predicted its type and text boundary. In this case, evaluation will be scored as:

$$Precision = \frac{1\,correct\ guess}{4\,system\ guesses} * 100 = 25\%$$

$$Recall = \frac{1\,correct\ guess}{5\,entities\ to\ be\ extracted} * 100 = 20\%$$

$$F - score = 2 * \frac{0.25 * 0.2}{0.25 + 0.2} * 100 = 22.2\%$$

Lastly, ACE has more complex evaluation algorithm. It defines a parameterized weight for different entity type. For example, person type is worth 1 and location type is worth 0.5. It also defines penalty for false positive, missed entities and type errors. Due to its specific evaluation

measure, it is not intuitive to interpret its result and compare with other evaluation methodologies.

Among above three main evaluation methodologies, we used CONLL's 'exact-match' evaluation for performance scoring. That is, partial matches either entity type match or entity text boundary match are not considered as correct recognition and false positive errors are not penalised. A named entity which exactly matches type and text boundary is only scored as a correct prediction.

Appendix C

Alphabetical list of part-of-speech tags used in the Penn Treebank Project

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun

19.	PPP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb