Improving the Performance of a Named Entity Recognition System with Knowledge Acquisition

Myung Hee Kim and Paul Compton

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

Abstract. Named Entity Recognition (NER) is important for extracting information from highly heterogeneous web documents. Most NER systems have been developed based on formal documents, but informal web documents usually contain noise, and incorrect and incomplete expressions. The performance of current NER systems drops dramatically as informality increases in web documents and a different kind of NER is needed. Here we propose a Ripple-Down-Rules-based Named Entity Recognition (RDRNER) system. This is a wrapper around the machine-learning-based Stanford NER system, correcting its output using rules added by people to deal with specific application domains. The key advantages of this approach are 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 Rule approach, with low-cost rule addition improved the Stanford NER system's performance on informal web document in a specific domain to the same level as its state-of-the-art performance on formal documents.

Keywords: Ripple-Down Rules, Named Entity Recognition.

1 Introduction

The Web contains a vast amount of information mainly in natural language that has been increasing exponentially. Most Web documents are relatively unstructured, with considerable noise and they change dynamically; therefore it is important to develop tools to manage unstructured data on the Web. A number of Natural Language Processing (NLP) applications have been developed to reduce the amount of time necessary to find the desired information from the Web, including Web Information Extraction (WIE), Automatic Text Summarization (ATS) Information Retrieval (IR) and Question-Answering (QA) systems.

Named Entity Recognition (NER) is one of key tasks in these NLP applications [1, 2]. It automatically identifies proper names in text and classifies them into a set of categories such as persons, geographical locations, names of organizations, dates, times, amounts of money etc. NER has mainly adopted two approaches. One is referred to as knowledge-based using explicit resources like rules and gazetteers, which usually are handcrafted by experienced language experts [4]. 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 corpora of annotated text. [7]. 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 [8] and unsupervised learning techniques [9], which do not require annotated corpora.

Current NER systems are trained mainly on journalistic documents such as news articles. Consequently they have not been trained to deal with the informality of Web documents, resulting in dramatic performance drops on Web documents. For these reasons, some studies comment that NER is a major source of errors for Web Information Extraction (WIE) [10, 11]. Recent WIE systems [11, 15] have avoided the NER process and instead utilized only shallow features like part-of-speech (POS) tags and chunk-phrase tags for entity extraction and extraction pattern generation and then relied on the Web's redundancy to improve accuracy. This approach has limitations for less redundant informal Web documents such as blogs and comment. Thus, developing NER systems for the Web is important for efficient information extraction from informal Web documents.

There are a number of studies on Web-scale NER. One approach relies on large Web resources. The lack of annotated corpora for Web documents and the cost of creating annotated corpora to cover highly heterogeneous Web documents lead to semi-supervised learning (SSL), which relies on very large collections of the Web documents and information redundancy [9]. Another approach develops gazetteers from Web data. Some studies have focused on automatic extraction of known entities from the Web or Wikipedia to build gazetteers appropriate for the web [13, 14].

However, most of these studies focused on the Web's heterogeneity rather than its informality. While machine learning (ML) based approaches are good for scaling domain coverage and achieves state-of-the-art performance, they have critical limitations in interpreting and fixing specific errors as they are discovered. Particularly, it is difficult to overcome errors caused by the Web's informal characteristics. Further, while gazetteers, which were automatically extracted from Web resources, could improve the coverage of Web vocabulary, they contain high levels of noise, which confuses context analysis NER techniques. Riloff et al. [16] demonstrated that when noise is introduced in gazetteers, the performance of an NER system degrades rapidly.

Web-scale NER is a tough challenge due to following characteristics of the Web's informality.

Informal Writing Styles. Huge amounts of Web documents are written informally not following strict writing styles like journalistic text [3]. Many NER techniques rely on title and trigger words. For example, in journalistic texts, person names are generally preceded by titles (Mr., Dr.), organization names are usually followed by trigger words (Inc., Ltd.) and location names are often identified by keywords (mountain, city). As these markers are often absent in Web documents, NER techniques, relying on such indicators, do not work efficiently and cause a significant numbers of errors.

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

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

In order to tackle the above challenges, we propose a Ripple-Down Rules based Named Entity Recognition (RDRNER) system. The RDRNER system employs the Stanford NER system as a base system and applies the Ripple-Down Rules technique to deal with the Web's informality. The Ripple-Down Rules technique 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, so errors can be corrected whenever they are identified.

The underling idea of the RDRNER system is that it takes state-of-the-art performance from a machine learning technique but then corrects any errors due to the Web's informality. With the RDRNER system, the user creates rules when the classification result provided by the system is incorrect. Since 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. The average F-score (F1) of the Stanford NER [17], trained on the CoNLL03 shared task dataset is 90.8% [18] but dropped to 76.96% on our Web dataset. After 4 hours knowledge acquisition with 200 sentences, the RDRNER system improved this performance to 90.48% similar to the Stanford NER's best performance on formal documents. The RDRNER system described below achieved 92.12% precision and 88.68% recall after training for a Web subdomain and then testing on other sentences from the same subdomain.

It should be noted that Stanford NER system and the RDRNER systems have quite different roles. The Stanford NER system is intended to be applied to any text and. In contrast because the RDRNER system is designed to fix the errors that occur, it is intended be used in specific domains of interest. That is, the user looking to identify entities in a particular domain will write rules for the errors that occur in that domain. If the scope of the application domain expands, the user writes more rules if and when needed. The strength of the approach comes from the ease and speed for which new rules can be added. This may seem a fairly limited solution, but we suggest that in practice organizations generally require technology like NER to deal with specific application domains, so that rapidly developing rules to tune a general purpose system for a particular domain may be a very attractive solution.

Our contributions can be summarized as follows:

- We have developed an RDRNER system that 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.

- We have evaluated the state-of-the-art Stanford NER system on a Web dataset with fair level of Web's informality and categorized error sources that critically degrade performance.
- We have demonstrated how the RDRNER system handles informally written Web documents and improves the performance of the Stanford NER system on informal documents in a restricted domain to be equivalent to its best performance on formal documents

The remainder of this paper is structured as follows. Section 2 presents related work and section 3 formally defines the task and presents the Web's informality challenges. Section 4 explains our RDRNER system in detail, section 5 presents the experimental setting and results and section 6 discusses the results and future work.

2 Related Work

2.1 Web Scale Named Entity Recognition

Since a large hand-annotated corpus is usually expensive and has coverage limitations, semi-supervised or unsupervised learning adopts categorised lists of known entities (called *gazetteers*) with lookup-based methods to recognize named entities [19]. This method needs much less time and labor effort since gazetteers can be generated using automated or semi-automated techniques [9].

As many tagging systems utilise gazetteers, some research has focused on creating gazetteers automatically using web page sources [9] or Wikipedia [13]. While Mikheev et al. [20] have shown that such gazetteers did not necessarily result in increased NER performance, Nadeau et al. [19] used gazetteers in an unsupervised named entity recognizer, and outperformed some other methods on the MUC Named Entity test dataset. Kazama et al. [14] also achieved a 3% F-score improvement using Wikipedia-based gazetteers in their named entity recognizer.

Liu et al. [21] studied NER performance on tweets that contains high level of noise such as excessive informal abbreviations and short hand. They proposed a combination of a K-Nearest Neighbours classifier and Conditional Random Fields and showed the effectiveness of KNN and semi-supervised learning through extensive experiments.

2.2 Ripple-Down Rules (RDR)

The basic idea of RDR is that cases are processed by the knowledge based system and when the output is not correct or missing one or more new rules are created to provide the correct output for that case. The knowledge engineering task in adding rules is simply selecting conditions for the rule. The rule is automatically located in the knowledge base with new rules placed under the default rule node for newly seen cases, and exception rules located under the fired rules. The system also stores cornerstone cases, cases that triggered the creation of new rules. If a new rule is fired by any cornerstone cases, the cornerstones are presented to the expert to select further differentiating features for the rule or to accept that the new conclusions should apply to the cornerstone. Experience suggests this whole process takes at most a few minutes. A recent study of a large number of RDR knowledge bases used for interpreting diagnostic data in chemical pathology, showed from logs that the median time to add a rule was less than 2 minutes across 57,626 rules [22].

The RDR approach has also been applied to a range of NLP applications. For example, Pham et al. developed KAFTIE, an incremental knowledge acquisition framework to extract positive attributions from scientific papers [23] and temporal relations that outperformed machine learning algorithms [24]. Relevant to the work here, RDR Case Explorer (RDRCE) [25] combined Machine Learning and manual Knowledge Acquisition for NLP problems. It automatically generated an initial RDR tree using transformation-based learning, but then allowed for corrections to be made. They applied RDRCE to POS tagging and achieved a slight improvement over state-of-the-art POS tagging after 60 hours of KA. We have recently demonstrated improved open information extraction using Ripple-Down Rules [26]. In this work we used the Stanford NER system and the limitations we found led to the work described here.

The idea of using an RDR system as a wrapper around a more general system was suggested by work on detecting duplicate invoices where the RDR system was used to clean up false positive duplicates from the general system [28].

3 The Web's Informality and NER

In this section, we illustrate the performance drop of the Stanford NER system on a Web dataset and categorise the type of NER errors that occur due to the Web's informality.

Table 1. The performance of the Stanford NER system on a Web dataset

	Person	Organization	Location	Money	Date	Time	Percent	All
P	90.4%	86.2%	84.4%	97.2%	87.2%	100%	100%	87.1%
R	75.3%	62.5%	81.4%	85.4%	85.0%	100%	66.7%	69.2%
F1	81.8%	72.7%	82.5%	90.6%	86.0%	100%	80.2%	77.0%

Table1 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 5.1. CONLL evaluation methodologies were used for the experiment. Overall, the Stanford NER system achieved a 77.0% F1 score on the Web data, while it achieved state-of-the-art 90.8% F1 score on the CONLL corpus [18]. This is about 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 the informal Web documents. The difference between precision and recall is around 18%.

Table 2. The Stanford NER system's error sources on the Web dataset

New vocabulary	39.53%
Machine learning inconsistency	25.97%
Informal capital letter usage	21.71%
Lack of trigger word	10.08%
Web noise	2.71%

Table 2 presents the causes of error for the Stanford NER system on the Web dataset. Error sources were classified into five categories: new vocabulary, ML inconsistency, informal capital letter usage, lack of trigger word and Web noise.

39.53% of errors were caused by new vocabulary that had not seen during training and were not contained in dictionaries used by the system. As noted, in order to solve this problem, some research has focused on creating gazetteers automatically from the Web [9] or Wikipedia [13] increasing the size of dictionary. Since the size of these Web gazetteers is quite large, most systems take a 'bag of words' approach that can cause confusion as its size increases. Another problem is the uncontrolled noise contained in the Web gazetteers. Liu et al. [21] have shown that the noise contained in the Web gazetteers reduces the NER performance. 25.97% of the errors were caused by Machine-Learning (ML) producing inconsistent annotation. For example, in one short Web sentence, '(**/O** Encyclopedia**/ORG**)**/O** Kafka**/PER**', Kafka was annotated as a PERSON. However, in another short sentence, '(**/O** Almanac**/ORG** –**/O** People**/O**)**/O** Kafka**/LOC**', Kafka was tagged as a LOCATION, although the second sentence has the evidence of the word 'People'. It is difficult to understand why such errors are made. 21.71% of the error is caused by informal Web capital letter usage. On the Web, capital letters are often used informally in order to emphasize the content, but this causes critical errors for current NER systems. For example, in the sentence 'Google**/ORG** Acquires**/ORG** YouTube**/ORG**', because 'Acquires' started with capital letter, the system annotated all three token as a one ORGANIZATION NE tag. 10.08% of errors were caused by lack of trigger words expected such as 'Ltd., Dr. and city'. For instance, in a sentence 'Franz**/PER** Kafka**/PER** –**/O** Prague**/LOC** wax**/O** museum**/O**', 'wax museum' was not tagged as LOCATION NE because museum is not recognised as a sort of trigger word. 2.71% of errors were caused by Web noise such as spelling errors, various symbols, excessive abbreviation and informal short hand. 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**', 'Googld' was not tagged due to a spelling error.

4 RDRNER System

The RDR-based Named Entity Recognition (RDRNER) system shown in Figure 1 consists of three main components: preprocessor, Stanford NER system and RDR KB learner. In section 4.1, the implementation details of the three components are explained; the RDR rule syntax is described in section 4.2 and RDR KB construction demonstrated in section 4.3. RDRNER rule examples to handle the Web's informality are presented in section 4.4 and finally the user interface is shown in section 4.5.

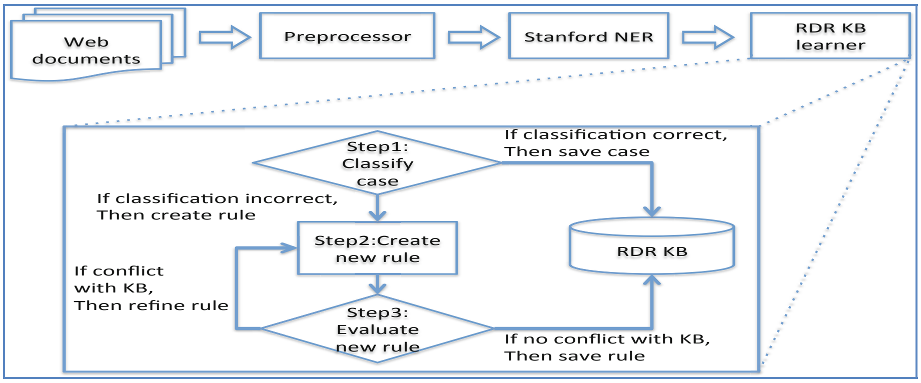

Web documents
Preprocessor
Stanford NER
RDR KB learner
Step1: Classify case
If classification correct, Then save case
If classification incorrect, Then create rule
Step2:Create new rule
RDR KB
If conflict with KB, Then refine rule
Step3: Evaluate new rule
If no conflict with KB, Then save rule

Fig. 1. Architecture of the RDRNER system

4.1 Implementation

Preprocessor. The preprocessor converts raw Web documents into a sequence of sentences, and annotates each token for Part-Of-Speech (POS) and noun and verb phrase chunk using the OpenNLP system. Annotated NLP features are utilized when creating RDR rules.

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 is based on the use of Gibbs sampling for inference in a Conditional Random Field (CRF) machine learning technique [17]. The system models were trained on data from CONLL, MUC6, MUC7, and ACE named-entity corpora. Each type of serialized classifier is provided with one more version that uses Clark's distributional similarity code to improve performance [27]. These versions of the classifier have additional features which provide better performance but require more memory. It achieves a 1.5% F-measure compared to the other versions. We used the classifier 'muc.7class.distsim.crf.ser.gz' that was trained on the MUC corpora and classifies 7 types of NE categories: PERSON, ORGANIZATION, LOCATION, DATE, TIME, MONEY, and PERCENT.

RDR KB Learner. The RDRNER KB 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 base NER system and adds rules when the classification results are not correct. There are three steps:

Step1: RDRNER classification
The RDR KB takes the given preprocessed sentence and the NER classification results from the Stanford NER system then returns RDRNER classification results. If RDR rules fired, the system replaces the Stanford NER system result with 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 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 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.

4.2 RDRNER's Rule Description

An RDR rule has a condition part and a conclusion part: 'IF (*condition*) THEN (*conclusion*)'. A *condition* consists of three components: (ATTRIBUTE, OPERATOR, VALUE). ATTRIBUTE refers to tokens of the given sentence. Currently the RDRNER system provides 8 types of OPERATOR as follows:

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

VALUE is derived automatically from the given sentence corresponding to the attribute and operator chosen by the user in the user interface. Conditions may connected with an 'AND' operation. A sequence of conditions using 'SEQ' operator is used to identify a group of words in sequence order, so patterns can be detected. For instance, the sequence condition '*IF SEQ(('Prague' hasNE 'LOC') AND ('wax' hasNE 'O') AND ('museum' hasNE 'O'))*' detects 'Prague**/LOC** wax**/O** museum**/O**'.

Rule's *conclusion* part has the following form:

```
(fixTarget,     // target word to amend NE classification
 fixType,       // amendment type (currently by default 'NE type' used)
 fixFrom,       // incorrect classification to be amended
 fixTo)         // correct classification which is amended
```

4.3 RDR KB Construction

The RDRNER system is based on Multiple Classification RDR (MCRDR) [5]. Figure 2 demonstrates MCRDR KB construction as the RDRNER system processes the following three cases starting with an empty KB.

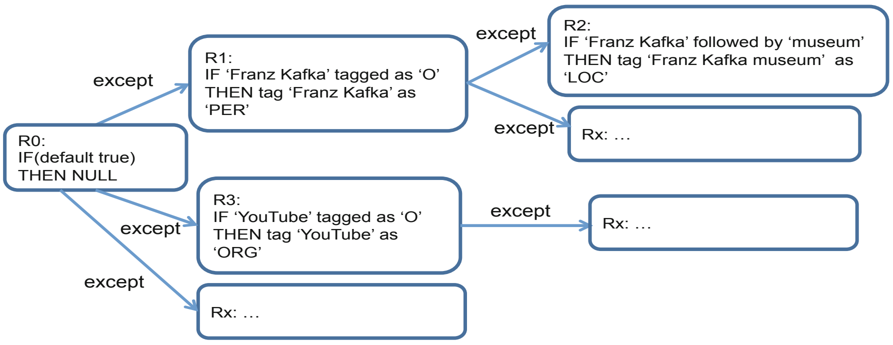

R0:
IF(default true)
THEN NULL
except
R1:
IF 'Franz Kafka' tagged as 'O'
THEN tag 'Franz Kafka' as
'PER'
except
R2:
IF 'Franz Kafka' followed by 'museum'
THEN tag 'Franz Kafka museum' as
'LOC'
except
Rx: …
except
R3:
IF 'YouTube' tagged as 'O'
THEN tag 'YouTube' as
'ORG'
except
Rx: …
except
Rx: …

Fig. 2. MCRDR structure of the RDRNER system

Case1: Franz/**O** Kafka/**O** was/**O** born/**O** into/**O** a/**O** Jewish/**O** family/**O** in/**O** Prague/**LOC** in/**O** 1883/**DATE**

1. The default rule R0 fired and the KB system returns a NULL classification so the Stanford NER output is not changed. The user decides this is incorrect as 'Franz Kafka' should be tagged as a PERSON NE.
2. A user adds a new rule R1 under the default rule R0.

Case2: Franz/**O** Kafka/**O** Museum/**O** (/**O** Easy/**ORG** Prague/**ORG** Travel/**ORG** Agency/**ORG** -/**O** Accommodation/**O** and/**O** Sightseeing/**O** in/**O** Prague/**LOC**)/**O**

1. Rule R1 fired but the user decides this is as an incorrect result because 'Franz Kafka Museum' should be tagged as an ORGANIZATION NE instead of tagging 'Franz Kafka' as a PERSON NE.
2. A user adds an exception rule R2 under the parent rule R1.

Case3: Google/**ORG** Buys/**O** YouTube/**O** for/**O** 1.65/**MONEY** Billion/**MONEY** Netscape.com/**O**

1. The default rule R0 fired and the KB system returns a NULL classification, which the user decides is an incorrect result as 'YouTube' is not tagged as an ORGANIZATION NE.
2. A user adds new rule R3 under the default rule R0.

This process continues with a new rule being added whenever the system does not give the correct classification.

4.4 RDR Rules to Handle the Web's Informality

In this section, we show examples of how the RDR rules handle five types of error: new vocabulary, ML inconsistency, informal capital letter usage, lack of trigger words and web noise, which were discussed in section 3.

The following is an example of an error caused by uncovered new vocabulary. The word 'YouTube' was not tagged by the Stanford NER system since it was not contained in the existing dictionary. A simple RDR rule is created to tag 'YouTube' as an ORGANIZATION NE that in effect extends 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** → NE classification error: 'YouTube' should be tagged as ORGANIZATION

RDR rule	*IF('YouTube' hasNE 'O')*
	THEN ('YouTube', 'NE type', '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 the Stanford NER system without obvious reason. A simple RDR rule is created to annotate 'Kafka' as a PERSON NE that amends the classification 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** → NE classification error: In case2, 'Kafka' is tagged as 'LOCATION'

RDR rule	*IF ('Kafka' hasNE 'LOC')*
	THEN ('Kafka', 'NE type', 'LOC', 'PER')

Resolved Case	(/**O** Almanac/**ORG** -/**O** People/**O**) /**O** Kafka/**PER**

The following is an example of an error due to informal capital letters. Because the word 'Acquire' started a capital letter, it was treated as part of an ORGANIZATION NE and resulted in an NE boundary error from the Stanford NER system. An RDR rule is created to 'Acquire' as a no NE which resolves the NE boundary error and correctly annotates two ORGANIZATION NE: Google and YouTube.

Error source	Informal capital letter usage
Problem Case	Google/**ORG** Acquire/**ORG** YouTube/**ORG** → NE boundary error: Because 'Acquire' is tagged as 'ORG', 'Google Acquire YouTube' treated as one ORGANIZATION NE tag instead of tagging 'Google' and 'YouTube' as ORGANIZATION separately

RDR rule	*IF ('Acquire' hasNE 'ORG')*
	THEN ('Acquire', 'NE type', 'ORG', 'O')

Resolved Case	Google/**ORG** Acquire/**O** YouTube/**ORG**

The following is an instance of an error caused by lack of trigger words in informal Web documents. Because there are no trigger words available to tag 'Prague wax museum' as one LOCATION NE, only 'Prague' was tagged as a LOCATION NE by the Stanford NER system. Two simple 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** → NE boundary error: ‘Prague wax museum’ should be tagged as one LOCATION NE

RDR rule	Rule1:
	IF SEQ((‘Prague’ hasNE ‘LOC’) AND (‘wax’ hasNE ‘O’) AND (‘museum’ hasNE ‘O’))
	THEN (‘wax’, ‘NE type’, ‘O’, ‘LOC’) AND (‘museum’, ‘NE type’, ‘O’, ‘LOC’)
	Rule2:
	IF ((‘Prague’ hasNE ‘LOC’) AND (‘Prague’ afterWD(+1) ‘wax’) AND (‘Prague’ afterWD(+2) ‘museum’))
	THEN (‘wax’, ‘NE type’, ‘O’, ‘LOC’) AND (‘museum’, ‘NE type’, ‘O’, ‘LOC’)

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

A more general rule for such a case might be that if a LOCATION NE is followed by the word museum within three words then the sequence is a location. This would implicitly recognise that museum in this context is a trigger word. It is entirely up to the person building rules whether they build specific or more general rules.

The following shows three examples of errors caused by noise. Case 1 shows ‘Googld’ not tagged due to a spelling error. Rule 1 is generated to tag the ‘Googld’ as an ORGANIZATION NE when there is the word ‘Google’ in the same sentence tagged as ORGANIZATION NE. Future conflict caused by this rule could be resolved by adding an exception rule under the rule as explained in section 4.3. Case 2 demonstrates that ‘Google’ was not tagged due to symbol ‘+’. Two simple rules (Rule 2 and Rule 3) are created to amend it. Case 3 presents a NE boundary error due to unknown abbreviation ‘Bn.’. Rule 4 is generated to fix the boundary error adding the ‘Bn.’ as part of MONEY NE when there is 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** → NE classification error: ‘Googld’ tagged as ‘O’ due to spelling error Case 2: symbols Google/**O** +/**O** YouTube/**O** :/**O** Day/**O** Two/**O** → NE classification error: ‘Google’ and ‘YouTube’ should be tagged as ORGANIZATION NE Case 3: abbreviations Google/**ORG** bought/**O** YouTube/**ORG** for/**O** $/**MONEY** 1.6/**MONEY** Bn./**O** Sweet/**O** → NE boundary error: ‘Bn.’ Should be tagged as a part of MONEY NE

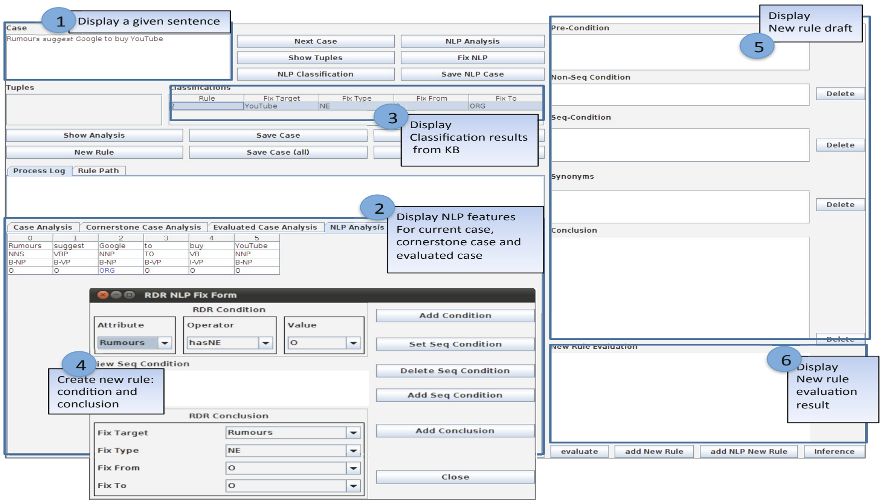
RDR rule	Rule 1 for case 1:
	IF((‘Google’ hasNE ‘ORG’) AND (‘Googld’ hasNE ‘O’))
	THEN (‘Googld’, ‘NE type’, ‘O’, ‘ORG’)
	Rule 2 and 3 for case 2:
	IF(‘Google’ hasNE ‘O’)
	THEN(‘Google’, ‘NE type’, ‘O’, ‘ORG’)
	IF(‘YouTube’ hasNE ‘O’)
	THEN(‘YouTube’, ‘NE type’, ‘O’, ‘ORG’)
	Rule 4 for case 3:
	IF((‘Bn.’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

Again, we do not suggest that these are ideal rules. The user can add whatever rules they like and the RDR approach simply ensures that the rules added do not degrade the performance of the knowledge base to date

4.5 RDRNER User Interface

The RDRNER system provides a graphic interface for creating and adding RDR rules and enabling the KB to be maintained by end-users. Because most of the relevant values are displayed automatically and the system is built based on the normal human process of identifying distinguishing features to justify a different conclusion, a user should be able to manage the system after few hours training. Industrial experience in complex domain supports this [22]. Figure 3 present the RDRNER user interface and the process flow.



Case
Rumours suggest Google to buy YouTube

Classifications

Rule	Fix Target	Fix Type	Fix From	Fix To
2	YouTube	NE	O	ORG

0	1	2	3	4	5
Rumours	suggest	Google	to	buy	YouTube
NNS	VBP	NNP	TO	VB	NNP
B-NP	B-VP	B-NP	B-VP	I-VP	B-NP
O	O	ORG	O	O	O

RDR Condition
Attribute: Rumours
Operator: hasNE
Value: O

View Seq Condition

RDR Conclusion
Fix Target: Rumours
Fix Type: NE
Fix From: O
Fix To: O

Fig. 3. User Interface of the RDRNER system

5 Experiments

Section 5.1 describes the Web dataset used. Section 5.2 shows the initial knowledge base construction of the RDRNER system and the section 5.3 present the results achieved by the RDRNER system and discusses how the system improved the existing performance of the Stanford NER system on the Web data for this particular domain. The CONLL evaluation methodology was used for experiments. The CONLL evaluation uses an 'exact-match' scoring methodology. That is, it only counts scores for named entity recognition, which satisfies both type classification and boundary detection.

5.1 Web Datasets

Web datasets were prepared from the MIL dataset developed by Bunescu et al. [6]. Bunescu et al. [6] 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. 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'. In the MIL dataset, each sentence has one pair of entities manually identified for their intended extraction task, but these 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 the 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 is widely used to evaluate Web Information Extraction (WIE) [11, 15]. We suggest that the MIL dataset represents a reasonable approximation to how an NER system would 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 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 data set. 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 entity types.

5.2 RDR Initial KB Constructions

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 the error 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 cases which received an incorrect classification result because of rules added earlier. In total, 83 rules were added within four hours. KB construction time covered from when a case is called up until a rule is accepted and this time is logged automatically. With the RDR approach this time covers all the time spent on knowledge acquisition, and no other time is spent validating or rule debugging outside of this.

We note that the approach may have introduced errors into cases correctly classified by the Stanford NER system. These would have been picked up and corrected with further rules if the normal RDR protocol had been used and cases were processed through both systems one by one. The effect of this is that there may be some errors in the test data which would have been corrected using the normal RDR protocol.

5.3 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 ORGANIZATION, 112 LOCATION, 41 MONEY, 40 DATE, 9 PERCENT, 2 TIME.

Table 3. The performance of the RDRNER system on each named entity class

	Person	Organization	Location	Money	Date	Time	Percent	All
P	95.1%	91.5%	86.4%	100%	89.7%	100%	100%	92.1%
R	92.0%	88.6%	84.8%	92.7%	87.5%	100%	88.9%	88.7%
F1	93.5%	90.5%	85.5%	96.4%	89.0%	100%	94.2%	90.5%

Table 3 presents the performance of the RDRNER system on seven NE classes on the test dataset. Overall, the RDRNER system achieved a 90.5% F1 score, while the Stanford NER system achieved a 77.0% F1 score on the same dataset (see table 1). That is, the RDRNER system improved the F1 score by 13.5% compared to the Stanford NER system. The 90.5% 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 [18]. On average, the RDRNER system achieved both high precision and recall. The difference between precision and recall is around 4%.

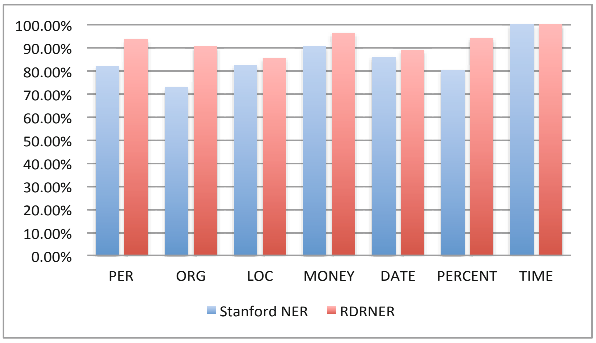


Fig. 4. Performance improvement of the RDRNER system over the Stanford NER system

Figure 4 presents the performance improvement of the RDRNER system over the Stanford NER system on each of the 7 NE classes in F1-score. For all classes the RDRNER system improved the Stanford NER performance except for 'TIME' where no rules were required as the Stanford NER system was already at 100%. ORGANIZATION and PERSON classes had the biggest improvement.

6 Discussion

Table 3, shows that the RDRNER system improves the Stanford system to give a 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 time 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 rapidly add rules to specifically cover the domain of interests. Such a solution may not be as elegant as an improved general purpose NER, but given the rapid knowledge acquisition time is certainly a practical solution for organisations needing higher NER in a domain of interest. The approach is not specific to the Stanford NER and can be used on top of better NER systems as they become available, and the better the general system the less knowledge acquisition time to tune it to the particular domain.

For NER, the use of lexical features such as dictionary and trigger words is a main technique. As discussed earlier, recent studies have focused on creating gazetteers automatically using Web sources, increasing the size of gazetteers dramatically. With the current 'bag of words' approach, the increased size of gazetteers cannot be utilised efficiently because word sense disambiguation problems. The lack of trigger words on the Web also make difficult to utilise lexical features by pre-defining trigger words.

The RDRNER system, however, can efficiently utilise lexical features for informal Web documents. Section 3 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, because rules are used any combination of conditions can be used to provide a much more sophisticated function than 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 new triggers, but again with a rule structure allowing a conjunction of conditions and exceptions to be used.

The RDRNER system is designed to be trained on a specific domain of interest. It would also be interesting to see if it was possible to extend the domain coverage to use crowd sourcing, whereby large numbers of people on the web might contribute rules, but there would major issues in how to combine such rules.

The RDRNER system required very little effort; rule creation is 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 [22]. 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, 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 system, one needs to keep monitoring cases until sufficient examples of each type of error have been accumulated for the learning system. We suggest it is simpler to correct the error when it first occurs so the user is then monitoring a continuously improving system. When errors are sufficiently infrequent, monitoring can stop.

References

1. Califf, M.E., Mooney, R.J.: Relational Learning of Pattern-Match Rules for Information Extraction. In: ACL 1997 Workshop in Natural Language Learning (1997)
2. Rozenfeld, B., Feldman, R.: Self-supervised relation extraction from the Web. Knowl. Inf. Syst. 17, 17–33 (2008)
3. Collot, M., Belmore, N.: Electronic Language: A New Variety of English. In: Computer-Mediated Communications: Linguistic, Social and Cross-Cultural Perspectives. John Benjamins, Amsterdam/Philadelphia (1996)
4. Rau, L.F.: Extracting Company Names from Text. In: 6th IEEE Conference on Artificial Intelligence Applications. IEEE Computer Society Press, Miami Beach (1991)
5. Kang, B.H., Compton, P., Preston, P.: Multiple Classification Ripple Down Rules: Evaluation and Possibilities. In: 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop (1995)
6. Bunescu, R.C., Mooney, R.J.: Learning to Extract Relations from the Web using Minimal Supervision. In: 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic (2007)
7. Asahara, M., Matsumoto, Y.: Japanese Named Entity Extraction with Redundant Morphological Analysis. In: Human Language Technology Conference - North American Chapter of the Association for Computational Linguistics (2003)
8. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Linguisticae Investigationes 30, 3–26 (2007)
9. Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., Yates, A.: Unsupervised named-entity extraction from the Web: An experimental study. Artif. Intell. 165, 91–134 (2005)
10. Nguyen, D.P.T., Matsuo, Y., Ishizuka, M.: Relation extraction from wikipedia using subtree mining. In: 22nd National Conference on Artificial Intelligence, vol. 2, pp. 1414–1420. AAAI Press (2007)
11. Zhu, J., Nie, Z., Liu, X., Zhang, B., Wen, J.R.: StatSnowball: a statistical approach to extracting entity relationships. In: 18th International Conference on World Wide Web, pp. 101–110. ACM, Madrid (2009)
12. Zacharias, V.: Development and Verification of Rule Based Systems — A Survey of Developers. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2008. LNCS, vol. 5321, pp. 6–16. Springer, Heidelberg (2008)
13. Toral, A., Muñoz, R.: 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 (2006)

14. Kazama, J.i., Torisawa, K.: ExploitingWikipedia as External Knowledge for Named Entity Recognition. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, Czech Republic (2007)
15. Banko, M., Etzioni, O.: The Tradeoffs Between Open and Traditional Relation Extraction. In: ACL 2008: HLT (2008)
16. Riloff, E., Jones, R.: Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In: 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence (1999)
17. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: The Association for Computer Linguistics (2005)
18. Ratinov, L., Roth, D.: Design Challenges and Misconceptions in Named Entity Recognition. In: CONLL 2009 (2009)
19. Nadeau, D., Turney, P.D., Matwin, S.: Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In: Lamontagne, L., Marchand, M. (eds.) Canadian AI 2006. LNCS (LNAI), vol. 4013, pp. 266–277. Springer, Heidelberg (2006)
20. Mikheev, A., Moens, M., Grover, C.: Named Entity recognition without gazetteers. In: 9th Conference on European Chapter of the Association for Computational Linguistics, pp. 1–8. Association for Computational Linguistics, Bergen (1999)
21. Liu, X., Zhang, S., Wei, F., Zhou, M.: Recognizing Named Entities in Tweets. In: 49th Association for Computational Linguistics, pp. 359–367 (2011)
22. Compton, P., Peters, L., Lavers, T., Kim, Y.S.: Experience with long-term knowledge acquisition. In: 6th International Conference on Knowledge Capture, pp. 49–56. ACM, Banff (2011)
23. Pham, S.B., Hoffmann, A.: Extracting Positive Attributions from Scientific Papers. In: Discovery Science Conference (2004)
24. Pham, S.B., Hoffmann, A.: Efficient Knowledge Acquisition for Extracting Temporal Relations. In: 17th European Conference on Artificial Intelligence, Riva del Garda, Italy (2006)
25. Xu, H., Hoffmann, A.: RDRCE: Combining Machine Learning and Knowledge Acquisition. In: Pacific Rim Knowledge Acquisition Workshop (2010)
26. Kim, M.H., Compton, P., Kim, Y.S.: RDR-based Open IE for the Web Document. In: 6th International Conference on Knowledge Capture, Banff, Alberta, Canada (2011)
27. Clark, A., Tim, I.: Combining Distributional and Morphological Information for Part of Speech Induction. In: 10th Annual Meeting of the European Association for Computational Linguistics (2003)
28. Ho, V.H., Compton, P., Benatallah, B., Vayssiere, J., Menzel, L., Vogler, H.: An incremental knowledge acquisition method for improving duplicate invoices detection. In: Proceedings of the International Conference on Data Engineering, Shanghai, China, pp. 1415–1418 (2009)