Domain Adaptation of Rule-Based Annotators for Named-Entity Recognition Tasks

Laura Chiticariu Rajasekar Krishnamurthy Yunyao Li Frederick Reiss Shivakumar Vaithyanathan
IBM Research – Almaden
650 Harry Road, San Jose, CA 95120, USA
{chiti, rajase, yunyaoli, frreiss}@us.ibm.com, shiv@almaden.ibm.com

Abstract

Named-entity recognition (NER) is an important task required in a wide variety of applications. While rule-based systems are appealing due to their well-known "explainability," most, if not all, state-of-the-art results for NER tasks are based on machine learning techniques. Motivated by these results, we explore the following natural question in this paper: *Are rule-based systems still a viable approach to named-entity recognition?* Specifically, we have designed and implemented a high-level language *NERL* on top of SystemT, a general-purpose algebraic information extraction system. *NERL* is tuned to the needs of NER tasks and simplifies the process of building, understanding, and customizing complex rule-based named-entity annotators. We show that these customized annotators match or outperform the best published results achieved with machine learning techniques. These results confirm that we can reap the benefits of rule-based extractors' explainability without sacrificing accuracy. We conclude by discussing lessons learned while building and customizing complex rule-based annotators and outlining several research directions towards facilitating rule development.

1 Introduction

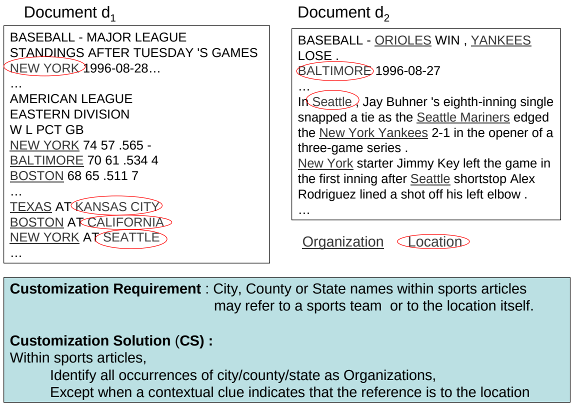
Named-entity recognition (NER) is the task of identifying mentions of rigid designators from text belonging to named-entity types such as persons, organizations and locations (Nadeau and Sekine, 2007). While NER over formal text such as news articles and webpages is a well-studied problem (Bikel et al., 1999; McCallum and Li, 2003; Etzioni et al., 2005), there has been recent work on NER over informal text such as emails and blogs (Huang et al., 2001; Poibeau and Kosseim, 2001; Jansche and Abney, 2002; Minkov et al., 2005; Gruhl et al., 2009). The techniques proposed in the literature fall under three categories: rule-based (Krupka and Hausman, 2001; Sekine and Nobata, 2004), machine learning-based (O. Bender and Ney, 2003; Florian et al., 2003; McCallum and Li, 2003; Finkel and Manning, 2009; Singh et al., 2010) and hybrid solutions (Srihari et al., 2001; Jansche and Abney, 2002).

1.1 Motivation

Although there are well-established rule-based systems to perform NER tasks, most, if not all, state-of-the-art results for NER tasks are based on machine learning techniques. However, the rule-based approach is still extremely appealing due to the associated transparency of the internal system state, which leads to better explainability of errors (Siniakov, 2010). Ideally, one would like to benefit from the transparency and explainability of rule-based techniques, while achieving state-of-the-art accuracy.

A particularly challenging aspect of rule-based NER in practice is domain customization — customizing existing annotators to produce accurate results in new domains. In machine learning-based systems, adapting to a new domain has traditionally involved acquiring additional labeled data and learning a new model from scratch. However, recent work has proposed more sophisticated approaches that learn a domain-independent base model, which can later be adapted to specific domains (Florian et

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1002–1012,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics



BASEBALL - MAJOR LEAGUE
STANDINGS AFTER TUESDAY 'S GAMES
NEW YORK 1996-08-28…
…
AMERICAN LEAGUE
EASTERN DIVISION
W L PCT GB
NEW YORK 74 57 .565 -
BALTIMORE 70 61 .534 4
BOSTON 68 65 .511 7
…
TEXAS AT KANSAS CITY
BOSTON AT CALIFORNIA
NEW YORK AT SEATTLE
…

BASEBALL - ORIOLES WIN , YANKEES LOSE .
BALTIMORE 1996-08-27
…
In Seattle , Jay Buhner 's eighth-inning single snapped a tie as the Seattle Mariners edged the New York Yankees 2-1 in the opener of a three-game series .
New York starter Jimmy Key left the game in the first inning after Seattle shortstop Alex Rodriguez lined a shot off his left elbow .
…

Customization Requirement : City, County or State names within sports articles may refer to a sports team or to the location itself.

Customization Solution (CS) :
Within sports articles,
Identify all occurrences of city/county/state as Organizations,
Except when a contextual clue indicates that the reference is to the location

Figure 1: Example Customization Requirement

```
// Core rules identify Organization and Location candidates

// Begin customization
// Identify articles covering sports event from article title
CR1 <SportsArticle> ← Evaluate Regular Expressions <R1>

// Identify locations in sports articles
CR2 Retain <Location> As <LocationMaybeOrg> If ContainedWithin <SportsArticle>

// City/County/State references (e.g., New York) may refer to the sports team in that city
CR3 Retain <LocationMaybeOrg> If Matches Dictionaries
      <'cities.dict','counties.dict','states.dict'>

// Some city references in sports articles may refer to the city (e.g., In Seattle )
// These references should not be reclassified as Organization
CR4 Discard <LocationMaybeOrg> If Matches Regular Expression <R2>
                                             on Left Context 2 Tokens

// City references to sports teams are added to Organization and removed from Location
CR5 Augment <Organization> With <LocationMaybeOrg>
// End customization


// Continuation of core rules
// Remove Locations that overlap with Organizations
     Discard <Location> If Overlaps Concepts <Organization>
```

Figure 2: Example Customization Rules in *NERL*

al., 2004; Blitzer et al., 2006; Jiang and Zhai, 2006; Arnold et al., 2008; Wu et al., 2009). Implementing a similar approach for rule-based NER typically requires a significant amount of manual effort to (a) identify the explicit semantic changes required for the new domain (e.g., differences in entity type definition), (b) identify the portions of the (complex) core annotator that should be modified for each difference and (c) implement the required customization rules without compromising the extraction quality of the core annotator. Domain customization of rule-based NER has not received much attention in the recent literature with a few exceptions (Petasis et al., 2001; Maynard et al., 2003; Zhu et al., 2005).

1.2 Problem Statement

In this paper, we explore the following natural question: Are rule-based systems still a viable approach to named-entity recognition? Specifically, (a) *Is it possible to build, maintain and customize rule-based NER annotators that match the state-of-the-art results obtained using machine-learning techniques?* and (b) *Can this be achieved with a reasonable amount of manual effort?*

1.3 Contributions

In this paper, we address the challenges mentioned above by (i) defining a taxonomy of the different types of customizations that a rule developer may perform when adapting to a new domain (Sec. 2), (ii) identifying a set of high-level operations required for building and customizing NER annotators, and (iii) exposing these operations in a domain-specific NER rule language, *NERL*, developed on top of SystemT (Chiticariu et al., 2010), a general-purpose algebraic information extraction system (Sec. 3). *NERL* is specifically geared towards building and customizing complex NER annotators and makes it easy to understand a complex annotator that may comprise hundreds of rules. It simplifies the identification of what portions need to be modified for a given customization requirement. It also makes individual customizations easier to implement, as illustrated by the following example.

Suppose we have to customize a domain-independent rule-based NER annotator for the CoNLL corpus (Tjong et al., 2003). Consider the two sports-related news articles in Fig. 1 from the corpus, where city names such as '*New York*' or '*Seattle*' can refer to either a *Location* or an *Organization* (the sports team based in that city). In the domain-independent annotator, city names were always identified as *Location*, as this subtle requirement was not considered during rule development. A customization to address this issue is shown in Fig. 1, which can be implemented in *NERL* with five rules (Fig. 2). This customization (explained in detail in Sec. 3) improved the $F_{\beta=1}$ score for *Organization* and *Location* by approximately 9% and 3%, respectively (Sec. 4).

We used *NERL* to customize a domain-independent rule-based NER annotator for three different domains – CoNLL03 (Tjong et al., 2003), Enron (Minkov et al., 2005) and ACE05 (NIST, 2005). Our experimental results (Sec. 4.3) demonstrate that the customized annotators have extraction quality better than the best-known results for

	Affects Single Entity Type	Affects Multiple Entity Types
Identify New Instances	C_S, C_{DD}, C_{DSD}	B_R
Modify Existing instances	C_{EB}, C_{DD}	C_{ATA}, C_G

Table 1: Categorizing NER Customizations

individual domains, which were achieved with machine learning techniques. The fact that we are able to achieve such results across multiple domains answers our earlier question and confirms that we can reap the benefits of rule-based extractors' explainability without sacrificing accuracy.

However, we found that even using *NERL*, the amount of manual effort and expertise required in rule-based NER may still be significant. In Sec. 5, we report on the lessons learned and outline several interesting research directions towards simplifying rule development and facilitating the adoption of the rule-based approach towards NER.

2 Domain Customization for NER

We consider NER tasks following the broad definition put forth by (Nadeau and Sekine, 2007), formally defined as follows:

Definition 1 *Named entity recognition is the task of identifying and classifying mentions of entities with one or more rigid designators, as defined by (Kripke, 1982).*

For instance, the identification of proper nouns representing persons, organizations, locations, product names, proteins, drugs and chemicals are considered as NER tasks.

Based on our experience of customizing NER annotators for multiple domains, we categorize the customizations involved into two main categories as listed below. This categorization motivates the design of *NERL* (Sec. 3).

Data-driven (C_{DD}): The most common NER customization is data-driven, where the customizations mostly involve the addition of new patterns and dictionary entries, driven by observations from the training data in the new domain. An example is the addition of a new rule to identify locations from the beginning of news articles (e.g., "BALTIMORE 1995-08-27" and "MURCIA , Spain 1996-09-10").

Application-driven: What is considered a valid named entity and its corresponding type can vary across application domains. The most common dimensions on which the definition of a named entity can vary are:

Entity Boundary (C_{EB}): Different application domains may have different definitions of where the same entity starts or ends. For example, a *Person* may (CoNLL03) or may not (Enron) include generational markers (e.g. "Jr." in "Bush Jr." or "IV" in "Henry IV").

Ambiguous Type Assignment (C_{ATA}): The exact type of a given named entity can be ambiguous. Different applications may assign different types for the same named entity. For instance, all instances of "White House" may be considered as *Location* (CoNLL03), or be assigned as *Facility* or *Organization* based on their context (ACE05). In fact, even within the same application domain, entities typically considered as of the same type may be assigned differently. For example, given "New York beat Seattle" and "Ethiopia beat Uganda", both '*New York*' and '*Ethiopia*' are teams referred by their locations. However, (Tjong et al., 2003) considers the former, which corresponds to a city, as an *Organization*, and the latter, which corresponds to a country, as a *Location*.

Domain-Specific Definition (C_{DSD}): Whether a given term is even considered a named entity may depend on the specific domain. As an example, consider the text "Commercialization Meeting - SBeck, BHall, BSuperty, TBusby, SGandhi-Gupta". Informal names such as '*SBeck*' and '*BHall*' may be considered as valid person names (Enron).

Scope(C_S): Each type of named entity usually contains several subtypes. For the same named entity task, different applications may choose to include different sets of subtypes. For instance, roads and buildings are considered part of *Location* in CoNLL03, while they are not included in ACE05.

Granularity(C_G): Name entity types are hierarchical. Different applications may define NER tasks at different granularities. For instance, in ACE05, *Organization* and *Location* entity types were split into four entity types (*Organization*, *Location*, *Geo-Political Entity* and *Facility*).

The different customizations are summarized as shown in Tab. 1, based on the following criteria: (i) whether the customization identifies new instances or modifies existing instances; and (ii) whether the customization affects single or multiple entities. For instance, C_S identifies new instances for a single entity type, as it adds instances of a new subtype for an existing entity type. Note that B_R in the table denotes the rules used to build the core annotator.

3 Named Entity Rule Language

3.1 Grammar vs. Algebraic NER

Traditionally, rule-based NER systems were based on the popular CPSL cascading grammar specification (Appelt and Onyshkevych, 1998). CPSL is designed so that rules that adhere to the standard can be executed efficiently with finite state transducers. Accordingly, the standard defines a rigid left-to-right execution model where a region of text can be matched by at most one rule according to a fixed rule priority, and where overlapping annotations are disallowed in the output of each grammar phase.

While it simplifies the design of CPSL engines, the rigidity of the rule matching semantics makes it difficult to express operations frequently used in rule-based information extraction. These limitations have been recognized in the literature, and several extensions have been proposed to allow more flexible matching semantics, and to allow overlapping annotations (Cunningham et al., 2000; Boguraev, 2003; Drozdzynski et al., 2004). However, even with these extensions, common operations such as filtering annotations (e.g. CR_4 in Fig. 2), are difficult to express in grammars and often require an escape to custom procedural code.

Recently, several declarative algebraic languages have been proposed for rule-based IE systems, notably AQL (Chiticariu et al., 2010) and Xlog (Shen et al., 2007). These languages are not constrained by the requirement that all rules map onto finite state transducers, and therefore can express a significantly richer semantics than grammar-based languages. In particular, the AQL rule language as implemented in SystemT (Chiticariu et al., 2010) can express many common operations used in rule-based information extraction without requiring custom code. In addition, the separation of extraction semantics from execution enables SystemT's rule optimizer and efficient runtime engine. Indeed, as shown in (Chiticariu et al., 2010), SystemT can deliver an order of magitude higher annotation throughput compared to a state-of-the-art CPSL-based IE system.

Since AQL is a general purpose information extraction rule language, similar to CPSL and JAPE, it exposes an expressive set of capabilities that go beyond what is required for NER tasks. These additional capabilities can make AQL rules more verbose than is necessary for implementing rules in the NER domain. For example, Fig. 3 shows how the same customization rule CR_4 from Fig. 2 can be implemented in JAPE or in AQL. Notice how implementing even a single customization may lead to defining complex rules (e.g. JAPE-R_1, AQL-R_1) and sometimes even using custom code (e.g. JAPE-R_2). As illustrated by this example, the rules in AQL and JAPE tend to be complex since some operations — e.g., filtering the outputs of one rule based on the outputs of another rule — that are common in NER rule sets require multiple rules in AQL or multiple grammar phases in JAPE.

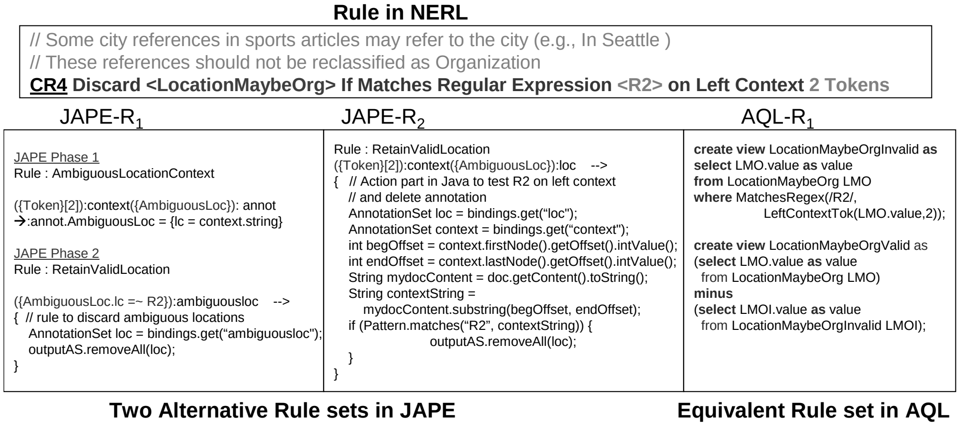
To make NER rules easier to develop and to understand, we designed and implemented Named Entity Rule Language (*NERL*) on top of SystemT. *NERL* is a declarative rule language designed specifically for named entity recognition. The design of *NERL* draws on our experience with building and customizing multiple complex NER annotators. In particular, we have identified the operations required in practice for such tasks, and expose these operations as built-in constructs in *NERL*. In doing so, we ensure that frequently performed operations can be expressed succinctly, so as not to complicate the rule set unnecessarily. As a result, *NERL* rules for named entity recognition tasks are significantly more compact and easy to understand than the equivalent AQL rules. At the same time, *NERL* rules can easily be compiled to AQL, allowing our NER rule development framework to take advantage of the capabilities of the SystemT rule optimizer and efficient runtime execution engine.

3.2 *NERL*

For the rest of this section, we focus on describing the types of rules supported in *NERL*. In Sec. 4, we shall demonstrate empirically that *NERL* can be successfully employed in building and customizing complex NER annotators.

A *NERL* rule has the following form:

$$\mathit{IntConcept} \leftarrow \mathit{RuleBody}(\mathit{IntConcept}_1, \mathit{IntConcept}_2, \ldots)$$



Rule in NERL

```
// Some city references in sports articles may refer to the city (e.g., In Seattle )
// These references should not be reclassified as Organization
CR4 Discard <LocationMaybeOrg> If Matches Regular Expression <R2> on Left Context 2 Tokens
```

JAPE-R_1

```
JAPE Phase 1
Rule : AmbiguousLocationContext

({Token}[2]):context({AmbiguousLoc}): annot
→:annot.AmbiguousLoc = {lc = context.string}

JAPE Phase 2
Rule : RetainValidLocation

({AmbiguousLoc.lc =~ R2}):ambiguousloc    -->
{  // rule to discard ambiguous locations
   AnnotationSet loc = bindings.get("ambiguousloc");
   outputAS.removeAll(loc);
}
```

JAPE-R_2

```
Rule : RetainValidLocation
({Token}[2]):context({AmbiguousLoc}):loc    -->
{   // Action part in Java to test R2 on left context
    // and delete annotation
    AnnotationSet loc = bindings.get("loc");
    AnnotationSet context = bindings.get("context");
    int begOffset = context.firstNode().getOffset().intValue();
    int endOffset = context.lastNode().getOffset().intValue();
    String mydocContent = doc.getContent().toString();
    String contextString =
       mydocContent.substring(begOffset, endOffset);
    if (Pattern.matches("R2", contextString)) {
                         outputAS.removeAll(loc);
    }
}
```

AQL-R_1

```
create view LocationMaybeOrgInvalid as
select LMO.value as value
from LocationMaybeOrg LMO
where MatchesRegex(/R2/,
              LeftContextTok(LMO.value,2));

create view LocationMaybeOrgValid as
(select LMO.value as value
  from LocationMaybeOrg LMO)
minus
(select LMOI.value as value
  from LocationMaybeOrgInvalid LMOI);
```

Two Alternative Rule sets in JAPE

Equivalent Rule set in AQL

Figure 3: Single Customization Rule expressed in *NERL*, JAPE and AQL

Intuitively, a *NERL* rule creates an intermediate concept or named entity (*IntConcept* for short) by applying a *NERL* rule on the input text and zero or more previously defined intermediate concepts.

***NERL* Rule Types** The types of rules supported in *NERL* are summarized in Tab. 2. In what follows, we illustrate these types by means of examples.

Feature definition (FD): FD rules identify basic features from text (e.g., *FirstName*, *LastName* and *CapsWord* features for identifying person names).

Candidate definition (CD): CD rules identify complete occurrences of the target entity. For instance, the Sequence rule "*LastName followed by ',' followed by FirstName*" identifies person annotations as a sequence of three tokens, where the first and third tokens occur in dictionaries containing last and first names.

Candidate Refinement (CR): CR rules are used to refine candidates generated for different annotation types. E.g., the *Filter* rule $CR3$ in Fig. 2 retains *LocationMaybeOrg* annotations that appear in one of several dictionaries.

Consolidation (CO): CO rules are used to resolve overlapping candidates generated by multiple CD rules. For instance, consider the text "Please see the following request from Dr. Kenneth Lim of the BAAQMD.". A CD rule may identify '*Dr. Kenneth Lim*' as a person, while another CD rule may identify '*Kenneth Lim*' as a candidate person. A consolidation rule is then used to merge these two annotations to produce a single annotation for '*Dr. Kenneth Lim*'.

***NERL* Examples** Within these categories, three types of rules deserve special attention, as they correspond to frequently used operations and are specifically designed to ensure compactness of the ruleset. In contrast, as discussed earlier (Fig. 3), each of these operations require several rules and possibly custom code in existing rule-based IE systems.

DynamicDict: The *DynamicDict* rule is used to create customized gazetteers on the fly. The following example shows the need for such a rule: While '*Clinton*' does not always refer to a person's last name (Clinton is the name of several cities in USA), in documents containing a full person name with '*Clinton*' as a last name (e.g., '*Hillary Clinton*') it is reasonable to annotate all references to the (possibly) ambiguous word '*Clinton*' as a person. This goal can be accomplished using the rule *<Create Dynamic Dictionary using Person with length 1 to 2 tokens>*, which creates a gazetteer on a per-document basis.

Filter: The *Filter* rule is used to discard/retain certain intermediate annotations based on predicates on the annotation text and its local context. Example filtering predicates include

- Discard C If Matches Regular Expression R
- Retain C If Contains Dictionary D on Local Context LC
- Discard C If Overlaps Concepts $C_1, C_2, \ldots$

ModifySpan: The *ModifySpan* rule is used to expand or trim the span of a candidate annotation. For instance, an *Entity Boundary* customization to include generational markers as part of a *Person* annotation can be implemented using a *ModifySpan* rule *<Expand Person Using Dictionary 'generation.dict' on RightContext 2 Tokens>*.

Using *NERL* Tab. 2 shows how different types of rules are used during rule building and customizations. Since B_R and C_S involve identifying one

Rule	*Category*	*Syntax*	B_R C_S	C_{DD} C_{DSD}	C_{EB}	C_G C_{ATA}
Dictionary	FD	**Evaluate Dictionaries** $< D_1, D_2, \ldots >$ with flags?	X	X		
Regex	FD	**Evaluate Regular Expressions** $< R_1, R_2, \ldots >$ with flags?	X	X		
PoS	FD	**Evaluate Part of Speech** $< P_1, P_2, \ldots >$ with language $< L >$?	X	X		
DynamicDict	FD	**Create Dynamic Dictionary** using *IntConcept* with flags?	X	X		
Sequence	CD	*IntConceptorString* multiplicity? (**followed by** *IntConceptorString* multiplicity?)+	X	X		
Filter	CR	**Discard/Retain** *IntConcept*(As *IntConcept*)? **If SatisfiesPredicate** on LocalContext	X	X		X
ModifySpan	CR	**Trim/Expand** *IntConcept* Using Dictionary $< D >$ on LocalContext	X		X	
Augment	CO	**Augment** *IntConcept* With *IntConcept*	X			X
Consolidate	CO	**Consolidate** *IntConcept* using ConsolidationPolicy	X			X

Table 2: Description of rules supported in *NERL*

or more entity (sub)types from scratch, all types of rules are used. C_{DD} and C_{DSD} identify additional instances for an existing type and therefore mainly rely on FD and CD rules. On the other hand, the customizations that modify existing instances (C_{EB},C_{ATA},C_G) require CR and CO rules.

Revisiting the example in Fig. 2, CR rules were used to implement a fairly sophisticated customization in a compact fashion, as follows. Rule CR_1 first identifies sports articles using a regular expression based on the article title. Rule CR_2 marks Locations within these articles as *LocationMaybeOrg* and Rule CR_3 only retains those occurrences that match a city, county or state name (e.g., '*Seattle*'). Rule CR_4 identifies occurrences that have a contextual clue confirming that the mention was to a location (e.g., '*In*' or '*At*'). These occurrences are already classified correctly as *Location* and do not need to be changed. Finally, CR_5 adds the remaining ambiguous mentions to *Organization*, which would be deleted from *Location* by a subsequent core rule.

4 Development and Customization of NER extractors with *NERL*

Using *NERL*, we have developed CoreNER, a domain-independent generic library for multiple NER extraction tasks commonly encountered in practice, including *Person*, *Organization*, *Location*, *EmailAddress*, *PhoneNumber*, *URL*, and *DateTime*, but we shall focus the discussion on the first three tasks (see Tab. 3 for entity definitions), since they are the most challenging. In this section, we first overview the process of developing CoreNER (Sec. 4.1). We then describe how we have customized CoreNER for three different domains (Sec. 4.2), and present a quality comparison with best published results obtained with state-of-the-art machine learning techniques (Sec. 4.3). The tasks we consider are not restricted to documents in a particular language, but due to limited availability of non-English corpora and extractors for comparison, our evaluation uses English-language text. In Sec. 5 we shall elaborate on the difficulties encountered while building and customizing CoreNER using *NERL* and the lessons we learned in the process.

4.1 Developing CoreNER

We have built our domain independent CoreNER library using a variety of formal and informal text (e.g. web pages, emails, blogs, etc.), and information from public data sources such as the US Census Bureau (Census, 2007) and Wikipedia.

The development process proceeded as follows. We first collected dictionaries for each entity type from different resources, followed by manual cleanup when needed to categorize entries collected into "strong" and "weak" dictionaries. For instance, we used US Census data to create several name dictionaries, placing ambiguous entries such as '*White*' and '*Price*' in a dictionary of *ambiguous last names*, while unambiguous entries such as '*Johnson*' and '*Williams*' went to the dictionary for *strict last names*. Second, we developed FD and CD rules to identify candidate entities based on the way named entities generally occur in text. E.g., <*Salutation CapsWord CapsWord*> and <*FirstName*

Type	Subtypes
PER	individual
LOC	Address, Boundary, Land-Region-Natural, Region-General, Region-International, Airport, Buildings-Grounds, Path, Plant, Subarea-Facility, Continent, Country-or-District, Nation, Population-Center, State-or-Province
ORG	Commercial, Educational, Government, Media, Medical-Science Non-Governmental

Table 3: NER Task Types and Subtypes

LastName> for *Person*, and <*CapsWord*{*1,3*} *OrgSuffix*> and <*CapsWord*{*1,2*} *Industry*> for *Organization*. We then added CR and CO rules to account for contextual clues and overlapping annotations (e.g., Delete *Person* annotations appearing within an *Organization* annotation).

The final CoreNER library consists of 104 FD (involving 68 dictionaries, 33 regexes and 3 dynamic dictionaries), 74 CD, 123 CR and 102 CO rules.

4.2 Customizing CoreNER

In this section we describe the process of customizing our domain-independent CoreNER library for several different datasets. We start by discussing our choice of datasets to use for customization.

Datasets For a rigorous evaluation of CoreNER's customizability, we require multiple datasets satisfying the following criteria: First, the datasets must cover diverse sources and styles of text. Second, the set of the most challenging NER tasks *Person*, *Organization* and *Location* (see Tab. 3) considered in CoreNER should be applicable to them. Finally, they should be publicly available and preferably have associated published results, against which we can compare our experimental results. Towards this end, we chose the following public datasets.

- CoNLL03 (Tjong et al., 2003): a collection of Reuters news stories. Consists of formal text.
- Enron (Minkov et al., 2005): a collection of emails with meeting information from the Enron dataset. Contains predominantly informal text.
- ACE05 (NIST, 2005)[1] a collection of broadcast news, broadcast conversations and newswire reports. Consists of both formal and informal text.

Customization Process The goal of customization is to refine the original CoreNER (hence referred to as CoreNER$_{orig}$) in order to improve its extraction quality on the training set (in terms of F$_{\beta=1}$) for each dataset individually. In addition, a development set is available for CoNLL03 (referred to as CoNLL03$_{dev}$), therefore we seek to improve F$_{\beta=1}$ on CoNLL03$_{dev}$ as well.

The customization process for each dataset proceeded as follows. First, we studied the entity definitions and identified their differences when compared with the definitions used for CoreNER$_{orig}$ (Tab. 3). We then added rules to account for the differences. For example, the definition of *Organization* in the CoNLL03 dataset contained a sports organization subtype, which was not considered when developing CoreNER. Therefore, we have used public data sources (e.g., Wikipedia) to collect and curate dictionaries of major sports associations and sport clubs from around the world. The new dictionaries, along with regular expressions identifying sports teams in sports articles were used for defining FD and CD rules such as CR_1 (Fig. 2). Finally, CR and CO rules were added to filter invalid candidates and augment the *Organization* type with the new sports subtype (similar in spirit to rules CR_4 and CR_5 in Fig. 2).

In addition to the train and development sets, the customization process for CoNLL03 also involved unlabeled data from the corpus as follows. 1) Since data-driven rules (C_{DD}) are often created based on a few instances from the training data, testing them on the unlabeled data helped fine tune the rules for precision. 2) CoNLL03 is largely dominated by sports news, but only a subset of all sports were represented in the train dataset. Using the unlabeled data, we were able to add C_{DD} rules for five additional types of sports, resulting in 0.31% improvement in F$_{\beta=1}$ score on CoNLL03$_{dev}$. 3) Unlabeled data was also useful in identifying domain-specific gazetteers by using simple extraction rules followed by a manual cleanup phase. For instance, for CoNLL03 we collected five gazetteers of organization and person names from the unlabeled data, resulting in 0.45% improvement in recall for CoNLL03$_{dev}$.

The quality of the customization on the train collections is shown in Tab. 5. The total number of rules added during customization for each of the three domains is listed in Tab. 4. Notice how rules of all four types are used both in the development

[1] The evaluation test set is not publicly available. Thus, following the example of (Florian et al., 2006), the publicly available set is split into a 80%/20% data split, with the last 20% of the data in chronological order selected as test data.

	FD	CD	CR	CO
CoreNER$_{orig}$	104	74	123	102
CoreNER$_{conll}$	179	56	284	71
CoreNER$_{enron}$	13	10	9	1
CoreNER$_{ace}$	83	35	117	26

Table 4: Rules added during customization

	Precision	**Recall**	**F$_{\beta=1}$**
CoreNER$_{conll}$	97.64	95.60	96.61
CoreNER$_{enron}$	91.15	92.58	91.86
CoreNER$_{ace}$	92.32	91.22	91.77

Table 5: Quality of customization on train datasets (%)

of the domain independent NER annotator, and during customizations for different domains. A total of 8 person weeks were spent on customizations, and we believe this effort is quite reasonable by rule-based extraction standards. For example, (Maynard et al., 2003) reports that customizing the ANNIE domain independent NER annotator developed using the JAPE grammar-based rule language for the ACE05 dataset required 6 weeks (and subsequent tuning over the next 6 months), resulting in improving the quality to 82% for this dataset. As we shall discuss shortly, *with similar manual effort, we were able to achieve results outperforming state-of-art published results on three different datasets, including* ACE05. However, one may rightfully argue that the process is still too lengthy impeding the widespread deployment of rule-based NER extraction. We elaborate on the effort involved and the lessons learned in the process in Sec. 5.

4.3 Evaluation of Customization

We now present an experimental evaluation of the customizability of CoreNER. The main goals are to investigate: (i) the feasibility of CoreNER customization for different application domains; (ii) the effectiveness of such customization compared to state-of-the-art results; (iii) the impact of different types of customization (Tab. 1); and (iv) how often different categories of *NERL* rules (Tab. 2) are used during customization.

We measured the effectiveness of customization using the improvement in extraction quality of the customized CoreNER over CoreNER$_{orig}$. As shown in Tab. 6, customization significantly improved

		Precision	**Recall**	**F$_{\beta=1}$**
CoNLL03$_{dev}$	CoreNER$_{orig}$	83.81	61.77	71.12
	CoreNER$_{conll}$	96.49	93.76	95.11
	Improvement	12.68	31.99	13.99
CoNLL03$_{test}$	CoreNER$_{orig}$	77.21	54.87	64.15
	CoreNER$_{conll}$	93.89	89.75	91.77
	Improvement	15.68	34.88	27.62
Enron	CoreNER$_{orig}$	85.06	69.55	76.53
	CoreNER$_{enron}$	88.41	82.39	85.29
	Improvement	3.35	12.84	8.76
ACE2005	CoreNER$_{orig}$	57.23	57.41	57.32
	CoreNER$_{ace}$	90.11	87.82	88.95
	Improvement	32.88	30.41	31.63

Table 6: Overall Improvement due to Customization (%)

			Precision	**Recall**	**F$_{\beta=1}$**
CoNLL03$_{dev}$	LOC	CoreNER$_{conll}$	**97.17**	95.37	**96.26**
		Florian	96.59	**95.65**	96.12
	ORG	CoreNER$_{conll}$	**93.70**	88.67	**91.11**
		Florian	90.85	**89.63**	90.24
	PER	CoreNER$_{conll}$	**97.79**	95.87	**96.82**
		Florian	96.08	**97.12**	96.60
CoNLL03$_{test}$	LOC	CoreNER$_{conll}$	**93.11**	91.61	**92.35**
		Florian	90.59	**91.73**	91.15
	ORG	CoreNER$_{conll}$	**92.25**	**85.31**	**88.65**
		Florian	85.93	83.44	84.67
	PER	CoreNER$_{conll}$	**96.32**	92.39	**94.32**
		Florian	92.49	**95.24**	93.85
Enron	PER	CoreNER$_{enron}$	**87.27**	**81.82**	**84.46**
		Minkov	81.1	74.9	77.9

Table 7: Comparison with state-of-the-art results(%)

F$_{\beta=1}$ score for CoreNER$_{orig}$ across all datasets. [2]

We note that the extraction quality of CoreNER$_{orig}$ was low on CoNLL03 and ACE05 mainly due to differences in entity type definitions. In particular, sports organizations, which occurred frequently in the CoNLL03 collection, were not considered during the development of CoreNER$_{orig}$, while in ACE05, *ORG* and *LOC* entity types were split into four entity types (*Organization*, *Location*, *Geo-Political Entity* and *Facility*). Customizations such as C_S and C_G address the above changes in named-entity type definition and substantially improve the extraction quality of CoreNER$_{orig}$.

Next, we compare the extraction quality of the

[2] CoNLL03$_{dev}$ and CoNLL03$_{test}$ correspond to the development and test sets for CoNLL03 respectively.

customized CoreNER for CoNLL03 and Enron[3] with the corresponding best published results by (Florian et al., 2003) and (Minkov et al., 2005). Tab. 7 shows that our customized CoreNER *outperforms the corresponding state-of-the-art numbers for all the NER tasks* on both CoNLL03 and Enron. [4] These results demonstrate that high-quality annotators can be built by customizing CoreNER$_{orig}$ using *NERL*, with the final extraction quality matching that of state-of-the-art machine learning-based extractors.

It is worthwhile noting that the best published results for CoNLL03 (Florian et al., 2003) were obtained by using four different classifiers (Robust Risk Minimization, Maximum Entropy, Transformation-based learning, and Hidden Markov Model) and trying six different classifier combination methods. Compared to the best published result obtained by combining the four classifiers, the individual classifiers performed between 2.5-7.6% worse for *Location*, 5.6-15.2% for *Organization* and 3.9-14.0% for *Person*[5]. Taking this into account, the extraction quality advantage of customized CoreNER is significant when compared with the individual state-of-the-art classifiers.

Impact of Customizations by Type. While customizing CoreNER for the three datasets, all types of changes described in Sec. 2 were performed. We measured the impact of each type of customization by comparing the extraction quality of CoreNER$_{orig}$ with CoreNER$_{orig}$ enhanced with all the customizations of that type. From the results for CoNLL03 (Tab. 8), we make the following observations.

- Customizations that identify additional subtypes of entities (C_S) or modify existing instances for multiple types (C_{ATA}) have significant impact. This effect can be especially high when the missing subtype appears very often in the new domain (E.g., over 50% of the organizations in CoNLL03 are sports teams).
- Data-driven customizations (C_{DD}) rely on the aggregated impact of many rules. While individual rules may have considerable impact on their own (e.g., identifying all names that appear as part of a player list increases the recall of PER by over 6% on both CoNLL03$_{dev}$ and CoNLL03$_{test}$), the overall impact relies on the accumulative effect of many small improvements.
- Certain customizations (C_{EB} and C_{DSD}) provide smaller quality improvements, both per rule and in aggregate.

	# rules added			Precision	Recall	F$_{\beta=1}$
C_{EB}	3	CoNLL03$_{dev}$	LOC	↑0.21	↑0.22	↑0.22
			ORG	↑1.35	↑0.38	↑0.59
			PER	-	-	-
		CoNLL03$_{test}$	LOC	↑0.30	↑0.36	↑0.33
			ORG	↑0.54	↑0.12	↑0.20
			PER	-	-	-
C_{ATA}	5	CoNLL03$_{dev}$	LOC	↑7.18	↓0.87	↑3.19
			ORG	↑1.37	↑10.67	↑9.04
			PER	↓0.04	-	↓0.01
		CoNLL03$_{test}$	LOC	↑7.73	↓1.20	↑3.77
			ORG	↑1.37	↑11.62	↑14.18
			PER	-	-	-
C_{DSD}	2	CoNLL03$_{dev}$	LOC	↑0.85	-	↑0.45
			ORG	↑1.00	↓0.07	↑0.01
			PER	-	-	-
		CoNLL03$_{test}$	LOC	↑0.04	↓0.12	↓0.12
			ORG	↑0.64	-	↑0.04
			PER	-	-	-
C_S	149	CoNLL03$_{dev}$	LOC	↑1.63	↓0.21	↑0.85
			ORG	↑11.44	↑40.79	↑39.73
			PER	↑0.13	-	↑0.05
		CoNLL03$_{test}$	LOC	↑3.71	↓0.18	↑2.05
			ORG	↑9.2	↑36.24	↑37.96
			PER	↑0.58	-	↑0.2
C_{DD}	431	CoNLL03$_{dev}$	LOC	↓0.94	↑10.18	↑3.99
			ORG	↑9.63	↑11.93	↑14.71
			PER	↑6.12	↑28.5	↑18.84
		CoNLL03$_{test}$	LOC	↓1.66	↑6.72	↑1.64
			ORG	↑8.84	↑12.40	↑15.90
			PER	↑9.15	↑31.48	↑22.21

Table 8: Impact by customization type on CoNLL03(%)

5 Lessons Learned

Our experimental evaluation shows that rule-based annotators can achieve quality comparable to that of state-of-the-art machine learning techniques. In this section we discuss three important lessons learned regarding the human effort involved in developing such rule-based extractors.

Usefulness of *NERL* We found *NERL* very helpful in that it provided a higher-level abstraction catered specifically towards NER tasks, thus hiding the complexity inherent in a general-purpose IE rule language. In doing so, *NERL* restricts the large space of operations possible within a general-purpose language to the small number of predefined "templates"

[3] We cannot meaningfully compare our results against previously published results for ACE05, which is originally used for mention detection while CoreNER considers only NER tasks.

[4] For Enron the comparison is reported only for *Person*, as labeled data is available only for that type.

[5] Extended version obtained via private communication.

listed in Tab. 2. (We have shown empirically that our choice of *NERL* rules is sufficient to achieve high accuracy for NER tasks.) Therefore, *NERL* simplifies development and maintenance of complex NER extractors, since one does not need to worry about multiple AQL statements or JAPE grammar phases for implementing a single conceptual operation such as filtering (see Fig. 3).

Is *NERL* Sufficient? Even using *NERL*, building and customizing NER rules remains a labor intensive process. Consider the example of designing the filter rule CR_4 from Fig. 3. First, one must examine multiple false positive *Location* entities to even decide that a filter rule is appropriate. Second, one must understand how those false positives were produced, and decide accordingly on the particular concept to be used as filter (*LocationMaybeOrg* in this case). Finally, one needs to decide how to build the filter. Tab. 9 lists all the attributes that need to be specified for a *Filter* rule, along with examples of the search space for each rule attribute.

Rule Attributes	Examples of Search Space
Location	Intermediate Concept to filter
Predicate Type	Matches Regex, Contains Dictionary, . . .
Predicate Parameter	Regular Expressions, Dictionary Entries, . . .
Context Type	Entity text, Left or Right context
Context Parameter	k tokens, l characters

Table 9: Search space explored while adding a *Filter* rule

This search space problem is not unique to filter rules. In fact, most rules in Tab. 2 have two or more rule attributes. Therefore, designing an individual *NERL* rule remains a time-consuming "trial and error" process, in which multiple "promising" combinations are implemented and evaluated individually before deciding on a satisfactory final rule.

Tooling for *NERL* The fact that *NERL* is a high-level language exposing a restricted set of operators can be exploited to reduce the human effort involved in building NER annotators by enabling the following tools:

Annotation Provenance Tools tracking provenance (Cheney et al., 2009) for *NERL* rules can help in explaining exactly which sequence of rules is responsible for producing a given false positive, thereby enabling one to quickly identify "misbehaved" rules. For instance, one can quickly narrow down the choices for the location where the filter rule CR_4 (Fig. 2) should be applied based on the provenance of the false positives. Similarly, tools for explaining false positives in the spirit of (Huang et al., 2008), are also conceivable.

Automatic Parameter Learning The most time-consuming part in building a rule often is to decide the value of its parameters, especially for FD and CR rules. For instance, while defining a CR rule, one has to choose values for the Predicate parameter and the Context parameter (see Tab. 9). Some parameter values can be learned – for example, dictionaries (Riloff, 1993) and regular expressions (Li et al., 2008).

Automatic Rule Refinement Tools automatically suggesting entire customization rules to a complex *NERL* program in the spirit of (Liu et al., 2010) can further reduce human effort in building NER annotators. With the help of such tools, one only needs to consider good candidate *NERL* rules suggested by the system without having to go through the conventional manual "trial and error" process.

6 Conclusion

In this paper, we described *NERL*, a high-level rule language for building and customizing NER annotators. We demonstrated that a complex NER annotator built using *NERL* can be effectively customized for different domains, achieving extraction quality superior to the state-of-the-art numbers. However, our experience also indicates that the process of designing the rules themselves is still manual and time-consuming. Finally, we discuss how *NERL* opens up several interesting research directions towards the development of sophisticated tooling for automating some of the rule development tasks.

References

D. E. Appelt and B. Onyshkevych. 1998. The common pattern specification language. In *TIPSTER workshop*.

A. Arnold, R. Nallapati, and W. W. Cohen. 2008. Exploiting feature hierarchy for transfer learning in named entity recognition. In *ACL*.

D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. In *Machine Learning*, volume 34, pages 211–231.

J. Blitzer, R. Mcdonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

B. Boguraev. 2003. Annotation-based finite state processing in a large-scale nlp arhitecture. In *RANLP*.

Census. 2007. U.S. Census Bureau. http://www.census.gov.

J. Cheney, L. Chiticariu, and W. Tan. 2009. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474.

L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. 2010. SystemT: An algebraic approach to declarative information extraction. In *ACL*.

H. Cunningham, D. Maynard, and V. Tablan. 2000. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield.

W. Drozdzynski, H. Krieger, J. Piskorski, U. Schäfer, and F. Xu. 2004. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 1:17–23.

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

J. R. Finkel and C. D. Manning. 2009. Nested named entity recognition. In *EMNLP*.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named entity recognition through classifier combination. In *CoNLL*.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *NAACL-HLT*.

R. Florian, H. Jing, N. Kambhatla, and I. Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *ACL*.

D. Gruhl, M. Nagarajan, J. Pieper, C. Robson, and A. Sheth. 2009. Context and domain knowledge enhanced entity spotting in informal text. In *ISWC*.

J. Huang, G. Zweig, and M. Padmanabhan. 2001. Information extraction from voicemail. In *ACL*.

Jiansheng Huang, Ting Chen, AnHai Doan, and Jeffrey F. Naughton. 2008. On the Provenance of Non-Answers to Queries over Extracted Data. *PVLDB*, 1(1):736–747.

M. Jansche and S. Abney. 2002. Information extraction from voicemail transcripts. In *EMNLP*.

J. Jiang and C. Zhai. 2006. Exploiting domain structure for named entity recognition. In *NAACL-HLT*.

Saul Kripke. 1982. *Naming and Necessity*. Harvard University Press.

G. R. Krupka and K. Hausman. 2001. IsoQuest Inc.: Description of the NetOwlTM extractor system as used for MUC-7. In *MUC-7*.

Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. V. Jagadish. 2008. Regular expression learning for information extraction. In *EMNLP*.

B. Liu, L. Chiticariu, V. Chu, H. V. Jagadish, and F. Reiss. 2010. Automatic Rule Refinement for Information Extraction. *PVLDB*, 3.

D. Maynard, K. Bontcheva, and H. Cunningham. 2003. Towards a semantic extraction of named entities. In *RANLP*.

A. McCallum and W. Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *CoNLL*.

E. Minkov, R. C. Wang, and W. W. Cohen. 2005. Extracting personal names from emails: Applying named entity recognition to informal text. In *HLT/EMNLP*.

D. Nadeau and S. Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26.

NIST. 2005. The ace evaluation plan.

F. J. Och O. Bender and H. Ney. 2003. Maximum entropy models for named entity recognition. In *CoNLL*.

G. Petasis, F. Vichot, F. Wolinski, G. Paliouras, V. Karkaletsis, and C. Spyropoulos. 2001. Using machine learning to maintain rule-based named-entity recognition and classification systems. In *ACL*.

T. Poibeau and L. Kosseim. 2001. Proper name extraction from non-journalistic texts. In *Computational Linguistics in the Netherlands*, pages 144–157.

E. Riloff. 1993. Automatically constructing a dictionary for information extraction tasks. In *KDD*.

S. Sekine and C. Nobata. 2004. Definition, dictionaries and tagger for extended named entity hierarchy. In *Conference on Language Resources and Evaluation*.

W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. 2007. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB*.

S. Singh, D. Hillard, and C. Leggeteer. 2010. Minimally-supervised extraction of entities from text advertisements. In *NAACL-HLT*.

P. Siniakov. 2010. *GROPUS - an adaptive rule-based algorithm for information extraction*. Ph.D. thesis, Freie Universitat Berlin.

R. Srihari, C. Niu, and W. Li. 2001. A hybrid approach for named entity and sub-type tagging. In *ANLP*.

E. F. Tjong, K. Sang, and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

D. Wu, W. S. Lee, N. Ye, and H. L. Chieu. 2009. Domain adaptive bootstrapping for named entity recognition. In *EMNLP*.

J. Zhu, V. Uren, and E. Motta. 2005. Espotter: Adaptive named entity recognition for web browsing. In *WM*.