ENTITY RECOGNITION FOR MULTI-MODAL SOCIO-TECHNICAL SYSTEMS

BY

AMIRHOSSEIN ALEYASEN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Advisers:

Assistant Professor Jana Diesner
Professor Emerita Marianne Winslett

Abstract

Entity Recognition (ER) can be used as a method for extracting information about socio-technical systems from unstructured, natural language text data. This process is limited by the set of entity classes considered in many current ER solutions. In this thesis, we report on the development of an ER classifier that supports a wide range of entity classes that are relevant for analyzing multi-modal, socio-technical systems. Another limitation with current entity extractors is that they mainly support the detection of named entities, typically in the form of proper nouns. The presented solution also detects entities not referred to by a name, such as general references to places (e.g. forest) or natural resources (e.g. timber). We use supervised machine learning for this project. To overcome data sparseness issues that results from considering a large number of entity classes, we built two separate classifiers for predicting labels for entity boundary and class. We herein investigate rules for merging both labels while minimizing the loss of accuracy due to this step. The accuracy of our classifier for the largest model with 94 classes achieves 75.9%. We compare the performance of our solution to other standard systems on several datasets, finding that with the same number of classes, the accuracy of our classifier is comparable to other state-of-the-art ER packages.

To my wife, for her love and support.

Acknowledgments

This work is supported by a start-up allocation award from the ”Extreme Science and Engineering Discovery Environment” (XSEDE). We also thank Brent Fegley (Illinois Informatics Institute, UIUC) and Joel Welling (Pittsburgh Supercomputing Center) for their help with developing the parallelized trainer, and Shubhanshu Mishra (The iSchool at Illinois, UIUC) for his help with model training.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information Extraction (IE) refers to the automatic extraction of structured information such as entities, attributes describing entities, and relationships between entities, from unstructured text sources. Annotating data using IE techniques enables richer queries over unstructured data than it is possible with keyword searches alone [22].

Named Entity Recognition (NER) is a subtask of IE that labels sequences of words, e.g. instances of persons, organizations, genes and proteins. In the expression "named entity", the word "named" aims to restrict the task to entities for which one or many rigid designators stands for the referent. Rigid designators include proper names such as *Abraham Lincoln*, and names for biological species and substances such as *sulfuric acid*.

NER based on noisy, unstructured sources is a challenging task that has engaged a large community of researchers from Natural Language Processing (NLP) and related fields, including machine learning, information retrieval, text mining, databases and data mining [16]. Extracting network data from text data is an example for a task that NER can be helpful for. In particular, researchers have used NER to locate entities that are then used as nodes for network construction. Finding actors and the geopolitical events that connect them is one example [10, 17, 23]. These data can also help to identify cultural factors and pieces of knowledge and information that affect the interaction between groups [15, 21, 29]. Organizing customer requests in customer services, building intra-organizational knowledge-bases, and creating a big picture of entities and their relation in bioinformatics articles are some other applications. Despite the high accuracy of some entity recognizers which are used in the aforementioned applications, there are some issues limiting their functionality.

In this thesis, we describe three of these issues and develop solutions to them.

First, the standard set of categories is usually limited to “people”, “organizations” and “locations”. For socio-technical systems, additional types such as events, goods, and manufactured or natural resources might be needed. In addition, these standard sets are usually limited to be referred to by a name, typically in the form of proper nouns. This limitation can be also problematic when there are entities in the text that are referred to by common nouns instead of proper nouns such as in news data [5]. For example, we might be interested in general references to places (e.g. forest), tasks (e.g. writing a thesis), resources (e.g. vehicles) or knowledge (e.g. expertise in data mining). To tackle these two issues, we develop an extractor for the more general task of identifying both named and unnamed examples of entities. We refer to this task with the broader term *“Entity Recognition”*. More formally, the goal with Entity Recognition (ER) is the identification and classification of instances of various entity classes in text data as efficiently and accurately as possible. In our solution, we apply supervised learning to previously annotated and validated data [31] and achieve accuracy rates of 76% to 79.8% for F1. The anticipated limitation with this solution is detecting more classes with lower accuracy. However, given the other methods used in the aforementioned areas (such as constructing look-up dictionaries and thesauri), our method is still efficient [8, 29].

The second issue is training time, which is not only important for experimentation, but also for retraining models in order to achieve domain adaptation and accounting for changes in the underlying annotation schema and ontologies. A related issue in any appropriate training data is the very low transition probabilities between entity classes. Typical solutions to this issue consider long-range dependencies, large feature spaces, and local plus global properties [11]. Conditional Random Fields (CRF) is a suitable learning technique given these constraints [4, 6, 9, 13]. However, the convex optimization in model training with CRFs requires significant memory, resulting in high training time. For example, training a model for all categories in our training data used to take about two weeks (on a Linux server with 64 GB memory and 16 quad core processors). To address this issue, in collaboration with Joel Welling from Pittsburgh Supercomputing Center, we parallelized one of the main yet serial CRF packages (Sarawagi), achieving better than a factor of 10 speed-up on 16 processors for our task. We will make the parallelized package publicly available.

The last issue that we cover in this thesis results from data spareness. Predicting the boundary and the class of an entity together in a join model suffers from sparseness. Our solution to this problem is building two separate classifiers for predicting the boundary and the class of an entity independently, and then combining their results. However, deciding for the best combination of labels in the case of inconsistencies between the classifiers is not an easy task. Therefore, we empirically tested various strategies and related error rates to develop a rule-based solution to this problem.

In summary, this thesis' contribution is three-fold:

- Training a classifier that predicts wide range of entity types which may or may not be referred to by a name (total of 94 classes).
- Exploiting a parallelized implementation to improve the training time by a factor of 10 speed-up in some cases.
- Building two classifiers and combining their results to mitigate the data sparseness issue.

In the next chapter, we discuss the background of entity recognition with respect to the given problem. In Chapter 3, we describe our method, explain the choice for a suitable training algorithm, the feature set, and the details of the implementation. In Chapter 4, we present the results and discuss them.

Chapter 2

Background

In 1991, entity recognition (ER) was introduced in a paper describing a system for "extracting and recognizing [company] names" [20]. Early extraction studies focused on the recognition of named entities, like people and company names, and relationship among them, from natural language text data.

New directions in ER have emerged based on recent research in areas like multimedia indexing and semi-supervised learning. For example, in multimedia indexing, there is an increasing interest in multimedia information processing (e.g., video, speech) where this information is extracted from text data [25]. In addition, the use of very large text collections triggered utilizing semi-supervised and unsupervised approaches for ER. Machine translation is another example of research areas that motivated new angles in ER [25].

In the first years of entity extraction, most of the studies depended on heuristics and handcrafted rules [16]. Therefore, they were able less to identify previously unknown entities. Most of the current research utilizes machine learning methods to automatically create rule-based systems or apply sequence labeling algorithms. These methods use a collection of training examples as a start [16].

After describing entity recognition in more detail, we explain the three types of machine learning methods that researchers usually use in ER. We then discuss features typically considered in entity recognition. Finally, we introduce standard evaluation methods for ER systems.

2.1 Entity Recognition

We can study ER from several perspectives including: language, textual genre or domain, and entity type. From the first perspective, a good proportion of work in ER uses text data in English data. The genre and domain of text data can have an effect on the performance of ER methods. We can consider different genres such as journalistic, scientific and informal writing; also different domains such as business, computer science and sports. Few studies are specifically devoted to diverse genres and domains. A system that is designed for a specific genre and domain may not work as well on other genres and domains. For instance, Poibeau and Kosseim [18] tested some systems on both the MUC-6 collection, which is composed of newswire texts, and on a proprietary corpus of manual translations of phone conversations and technical emails. They report a drop in performance for every system (about 20% to 40% of precision and recall)[16].

Another major limitation of ER systems is the scope of the supported entity types. Early work formulates the ER problem as recognizing “proper names”. Overall, the most studied types are names of persons, locations and organizations. A recent interest in bioinformatics and the availability of the bio corpora led to many studies dedicated to alternative entity types such as proteins, DNA, RNA, cell types, drugs and chemicals [27, 26, 28].

Other work considers a much wider set of entities. In this line of research a named entity hierarchy that includes many fine grained subcategories defined is usual. These systems support a wide range of categories such as museum, river, airport, product, event, substance, animal, religion, or color. The number of categories is about 200 in some cases [24].

2.2 Learning Methods

2.2.1 Supervised Learning

In supervised learning (SL), a large collection of annotated documents is used to study the features of positive and negative examples of named entities to capture instances of a given type. Hidden Markov Models (HMM), Conditional Random Fields (CRF), Decision Trees, Maximum Entropy Models (ME) and Support Vector Machines (SVM) are the main methods for supervised learning [25].

Hidden Markov Model (HMM) is an instance of generative models. They estimate a joint distribution of the form $P(x, y, ...)$. As an example, Bikel et al. [2] employed HMM in speech recognition and named entity recognition. Their system, called IdentiFinder, applied multiple word features, and obtained an accuracy of up to 94.9% for limited types (person, organization, location).

Conditional models, as an alternative to generative models, estimate a conditional distribution of the form $P(y|x)$. That is, conditional models look for the most likely sequence of class labels y given an observed sequence of x (such as a sentence) [9].

The main superiority of conditional models compared to generative models is enabling the usage of arbitrary features of the x's, such as global and long-distance features [6]. In conditional models, information related to distant classes (e.g. entity types) can communicate directly in the model. Conditional Random Fields (CRF) [9] is a conditional model that used widely for NER and that has achieved higher accuracy rates than generative models.

Despite of the mentioned advantages of SL, its need for a large annotated corpus is a major weakness. The lack of access to such resources as well as the excessively high cost of building them resulted in two other learning approaches: semi-supervised learning and unsupervised learning, that are discussed in next sections.

2.2.2 Semi-supervised Learning

Semi-supervised learning (SSL) methods employ Bootstrapping and a set of seeds as the main approach to begin the learning process. Consider a system aimed at a particular entity type. It might start by asking the user to specify a small set of example names. The system then proceeds to find sentences containing these names. It uses these sentences to recognize contextual clues common to the examples. Afterwards, by reapplying the learning process to the newly found examples, it tries to find new relevant contexts. Through this repetition, the system will collect a large number of the entity type names and contexts [16].

2.2.3 Unsupervised Learning

Unsupervised learning (UL) usually relies on clustering. For example, the similarity of context can be used to collect named entities from clustered groups. There are also other approaches in unsupervised learning that basically depend on lexical resources (e.g., WordNet [14]), lexical patterns, and statistics computed on large unannotated corpora [25].

2.3 Feature Space in Entity Recognition

We group features in the three categories: word-level features, list lookup features, and document and corpus features [16].

Word-level features: Word-level features are related to the character makeup of words. They specifically describe word case, punctuation, numerical values and special characters. Examples of word-level features are “starts with a capital letter”, “ends with period”, “has internal period”, morphology (such as prefix, suffix, singular version, stem), token length and phrase length.

List lookup features: Lists (or gazattees, dictionaries, lexicons) are one feature in NER. List inclusion is a way to express the relation “is a” (e.g., Champaign is a city). It may appear obvious that if a word (Champaign) is an element in a list of cities, then the probability of this word to be city, in a given text, is high. However, because of word polysemy, the probability is almost never 1. We can used different sources for providing list lookup features such as general lists (dictionaries, stop words, capitalized nouns or common abbreviations), list of entities (airlines, celebrities, astral bodies) and list of entity cues (such as typical words in organizations, person titles, name prefixes). There are many publicly available knowledge base (e.g. Freebase [3], DBPedia [1]) that are the rich sources for lookup features.

Document and corpus features: Document features are defined by both document content and document structure. Large collections of documents (corpora) are also suitable sources of features. These features go beyond the single word and multiword expressions and include meta-information about documents as well as corpus statistics. Examples of document and corpus features are multiple occurrences (other entities in the same context, uppercased and lowercased occurrences, anaphora, coreferences), local syntax (enumeration, apposition, position in sentence, in paragraph, and in document) meta information (URL, email header, XML section, bulleted/numbered lists, tables, figures) and corpus frequency (word and phrase frequency, co-occurrences, multiword unit permanency)[16].

2.4 Evaluation of Entity Recognition

The scope of Entity Recognition research was strongly influenced by two competitions, the Message Understanding Conference (MUC) and Automatic Content Extraction (ACE) program. Since evaluation of ER systems is essential to progress in that field, these communities proposed many techniques to rank systems based on their capability to annotate a text like an expert linguist. There exist different main scoring techniques used for the MUC, IREX, CONLL and ACE conferences. Generally in NER, systems are usually evaluated based on how their output compares to gold standard data [16].

For instance, in MUC, a system is scored on two criteria: its ability to find the correct type (TYPE) and exact text (TEXT). A correct TYPE is credited if an entity is assigned the correct type, regardless of boundaries, as long as there is an overlap. A correct TEXT is credited if entity boundaries are correct, regardless of the type. The final MUC score is the micro-averaged F-measure (MAF), which is the harmonic mean of precision and recall calculated over all entities on both criteria [25].

On the other side, IREX and CONLL share a simple scoring system called exact-match evaluation.

$$P = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FP_i} \tag{2.1}$$

$$R = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FN_i} \tag{2.2}$$

$$F = \frac{2PR}{P + R} \tag{2.3}$$

In the exact-match evaluation, systems are compared based on the micro-averaged F-measure (MAF). Using MAF, the individual true positives (TP), false positives (FP), and false negatives (FN) for different categories used for precision (P), recall (R) and F-measure (F) calculations.

Chapter 3

Method

In order to select an appropriate learning technique, the characteristics of the training data need to be considered. The training data for ER tasks are often sparse. This means that even with a large annotated text corpus, only a small portion of the data are entities of interest, while the vast majority of words is irrelevant [12].

The next issue is the sequential nature of human language. This means, the words that construct sentences and text are not drawn independently from a distribution and they have significant sequential correlation. Thus, an appropriate learning technique should consider this characteristic to exploit the available information as effective as possible. So, sequential learning seems appropriate for this purpose. One of the most effective sequential learning method based on these constrains is CRF. However, there are some other methods such as HMM and MEMM, but due to their limitations, they are not an appropriate candidate for sparse data.

3.1 Data

There are many annotated dataset that researchers can use for training their desired classifiers. Two of them are ACE2005 [30] and BBN [31] that are well-known for constructing models suitable to answering questions in the mentioned fields. Both datasets make a distinction between specific (named) versus generic (mainly referred to by common nouns) instances. Since after discounting pronouns, BBN contains more instances (N= 171,877) in comparison to ACE (N=12,318), we use BBN for training. The dataset had a few XML inconsistency issues. After fixing these issues, we partitioned the data into k=5 folds for training and cross-validation. Table 3.1 shows the entity classes available in BBN that considered for learning.

3.2 Model Training

We train four class label models based on Table 3.1:

- Main type (column 1) (cat, 10 classes)
- Main type plus subtype (columns 1+2) (catsub, 30 classes)
- Main type plus specific versus generic (columns 1+3) (catspec, 15 classes)
- Main type plus subtype plus specificity (columns 1+2+3) (model 4, 94 classes)

Each model or level of detail might be useful for different tasks or meeting different accuracy needs. For example, the subtype of a node (model 2) can be used as node attributes, while model 3 is useful for partitioning graphs into specific versus generic social entities and using them as the unit of analysis.

We also need to correctly locate each entities' boundaries. There are three models for determining the boundaries.

- BIO (begin, inside, outside)
- BIEO (begin, inside, end, other)
- BIEOU (begin, inside, end, other, unigram)

Ratinov and Roth [19] showed that BIEOU outperforms BIO by 0.5% to 1.3%. We therefore choose this model.

One of the advantage of using CRF is supporting arbitrarily large numbers of features, such as word feature to consider long-distance information. Thus, CRFs involve high time

Table 3.1: Entity Classes (BBN)

Main Type	**Subtype**	**Specific (s) / Generic (g)**
Person	-	both
Organization	Corporation	both
	Educational	both
	Government	both
	Attractions	both for hospital, hotel, museum, other
	Political	both
	Religious	both
NORP	Religion	s
	Nationality	s
	Other	s
	Political	s
Facility/ Location/ GPE	Facility	both for airport, attraction, bridge, building, highway /street, other
	City	both
	Country	both
	Location	s for other, border, continent, lake/sea/ocean, region, river
	State/province	both
Objects	Animal	no distinction
	Disease	no distinction
	Plant	no distinction
	Product	both for vehicle, weapon, other
	Substance	no distinction for chemical, drug, food, nuclear, other
	Money	no distinction
Intellectual Property	Language	s
	Law	s
	Work Of Art	s for book, painting, play, song, other
Event	Event	s for hurricane, other
	War	s
Game	-	no distinction
Date/Time	-	no distinction for date, duration, time, other
Quantities	Quantity	no distinction for cardinal, ordinal, percent, 1D, 2D, 3D, energy, other, speed, temperature, contact info phone, contact info other, weight
	Age	no distinction

complexity for training due to performing global search in a large feature space. We next address this problem by parallelizing the trainer, in collaboration with Joel Welling from Pittsburgh Supercomputing Center. It should be noted, inference is not subject to this constrain and it will happen in time efficient fashion.

3.3 Implementation

For CRF implementation we used the public available CRF package provided by Sarawagi [22]. This package that implemented in Java provides a basic implementation of a CRF that can be adjusted and customized for specific types of CRF applications. The package includes some predefined features that is given in Table 3.3. In addition, it has the capability to add new features.

Analogous to our definition of the Entity Extraction process, our CRF implementation consists of two steps: First, the CRF identifies relevant terms. These terms are marked as being a part of a relevant entity. If consecutive words are identified as belonging to one entity (e.g. World Food Programme), they are deterministically designated as one concept. Second, the CRF is used to classify the identified relevant entities. In order to analyze and evaluate the accuracy achieved by both steps, we measure and report accuracy rates for each step separately.

The Boundary Detector (BDec) detects the boundary of entities in text. It uses five different labels (begin, end, inside, unique and irrelevant) for annotating each word.

The Class Detector (CDec) predicts the classes of the entities for entity types given in Table 3.1. CDec trains a CRF model based on training data. It comes with an efficient feature extractor that supports a wide range of features. The features are organized into feature sets. A feature set combines one or more features used to train and test a single model. The list of feature sets given in Table 3.3.

Table 3.2: Features in the CRF Learning Model (Source: http://crf.sourceforge.net).

Feature ID	Name	Feature Description
ST	Start	This feature checks whether the current label can be a start state or not, and fires accordingly.
ED	End	This feature checks whether the current label can be an end state or not, and fires accordingly.
EG	Edge	This feature is transition feature, solely dependent upon current label and previous 'y' values. It encodes transition information and allows sequential information to be included in the model.
UN	Unknown	The feature fires when the current token is not observed in the training data.
WD	Word	The feature checks whether the current token is present in the dictionary in the particular state under consideration or not. The dictionary is created on-the-fly from the training set.
WS	WordScore	The feature returns log of the ratio of current word with the label y to the total words with label y.
RG	Regex	The feature checks several properties such as: isInitCapitalWord, isAllCapitalWord, isAllSmallCase, singleCapLetter, containsDashes, containsDash, singleDot, singleComma, singleQuote, isSpecialCharacter, fourDigits, isDigits, containsDigit, endsWithDot.

3.4 Feature Selection

We use the features given in Table 3.3. These features come with the base CRF package. We add two more features; parts of speech and look-up dictionary features. We will introduce them in next sections. Individual features are organized into feature sets. Different feature sets used for training the models. However some of the features are not highly predictive, but aggregating different features as a feature set enables more robust prediction, even if the individual contribution per feature might be weak.

3.5 Parallelization

For training the models, BDec and CDec modules support single-CPU and parallel execution. User can select number of CPUs based on their resources and requirements. In parallel execution, the tool uses a parallel architecture that executes several Java threads (on one or more CPUs) for training the CRFs model.

During the evaluation of the objective function, each training example contributes in a parallel fashion. The running time for each training task grows with the number of features in the training set, as this corresponds to the number of weights to be optimized. The test examples for this project were based on two feature sets, the boundary and category features. The category set includes 95 labels, while the Boundary set includes 5 labels. Thus training time is much longer for the Category problem. Since the Boundary problem runs in a few minutes, there is no great need to improve its performance. The category problem takes hours to run serially.

The results for two same parallel executions could differ slightly. It should be noted that the differences are minor and less than 0.01% on precision, recall and F-measure for most of the cases. The possible causes for different results given in the following:

- Since different threads handle different training cases and based on them, do many optimization steps, it is possible to have different results.

- Floating point math will give slightly different answers when the numbers are added in a different order, because of rounding

3.6 Additional Features

We also experimented with using parts of speech (POS) and lexical features, such as external dictionaries, as additional features.

To provide the POS feature, we preprocess the training data and extract the POS for each word using the Stanford POS Tagger[1].

To provide the lexical feature we use Freebase[2], which is a large collaborative knowledge base consisting of data provided mainly by its community members. A *topic* in Freebase represents a single concept or a real-world thing. A *type* denotes an *IS-A* relationship about a topic. For example, if the "Shakespeare" topic has the "Person" type it means that Shakespeare IS-A person. For each entity class in the BBN dataset, we manually find the corresponded type in Freebase and extract the list of topics for them as the lexical feature. We randomly select 5000 topics for the types that have more topics, such as location. We preprocess the training data and assign the topic for each word if it appears in at least one list. If the word appears in more than one list, we randomly select one of them.

In some cases, two different entities have words in common, for example, *New York* is a city, but *New York Institute of Technology* is an organization. In this case, since *New York* exists in the city list, using the lexical feature may be misleading for recognizing the organization. To resolve this problem, we assign topics to longer n-grams first, and if an n-gram appears in a list, we assign the topic to the n-grams and exclude it and all of its subsequences for further processing.

[1]http://nlp.stanford.edu/software/tagger.shtml
[2]https://www.freebase.com/

Chapter 4

Experimental Results

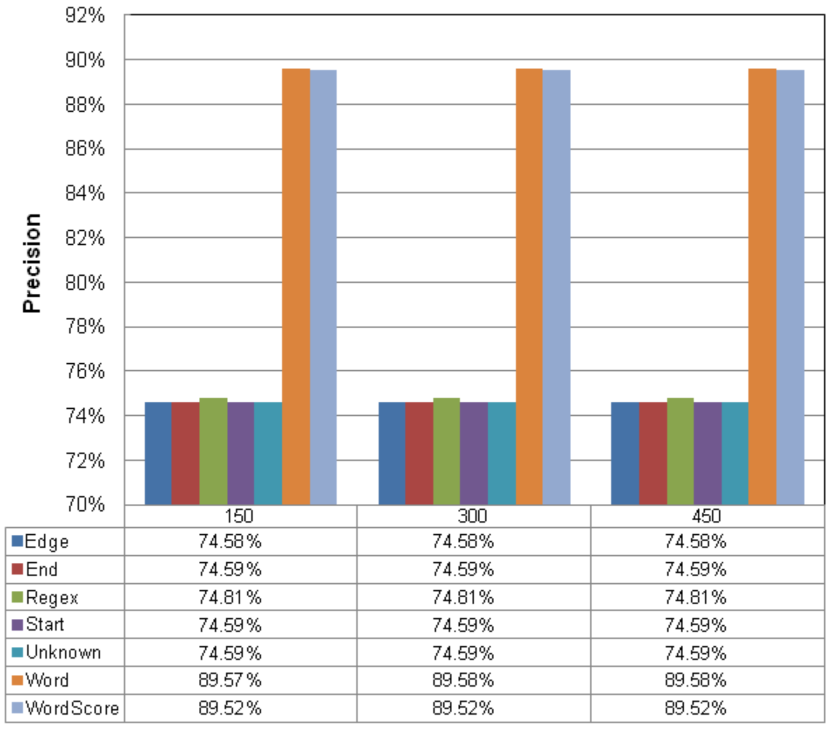
For evaluating the efficiency and effectiveness of the classifier, we ran different experiments. In this section, we explain the experiments that assess the accuracy of the BDec and CDec modules while testing different characteristics including parallelization, feature selection and number of iterations. In the all experiments, we used k-fold cross-validation (k=5) for determining precision, recall and the F-measure.

4.1 Number of Iterations

We measure the effect of increasing the number of iterations on the prediction. We performed boundary detection for 150, 300 and 450 iterations and use the one-feature strategy, which means in each experiment, we use just one feature. As shown in Figure 4.1, the precision for different numbers of iterations does not change. Thus, a higher number of iterations has no effect on the precision of the BDec method. The results are similar for recall and F-measure. For CDec, we conduct the same experiment (150, 300 and 450 iterations) finding similar results.

Table 4.1: Accuracy per class model

Model (# classes)	**Prec**	**Recall**	**F1**
Boundary (5)	93.46%	78.96%	85.60%
Cat (10)	86.68%	73.97%	79.82%
Catsub (30)	84.55%	72.68%	78.17%
Catspec (15)	84.99%	72.91%	78.49%
All (94)	81.52%	71.17%	75.99%



	150	300	450
🟦 Edge	74.58%	74.58%	74.58%
🟥 End	74.59%	74.59%	74.59%
🟩 Regex	74.81%	74.81%	74.81%
🟪 Start	74.59%	74.59%	74.59%
🟦 Unknown	74.59%	74.59%	74.59%
🟧 Word	89.57%	89.58%	89.58%
🟦 WordScore	89.52%	89.52%	89.52%

Figure 4.1: Effect of number of iterations on precision

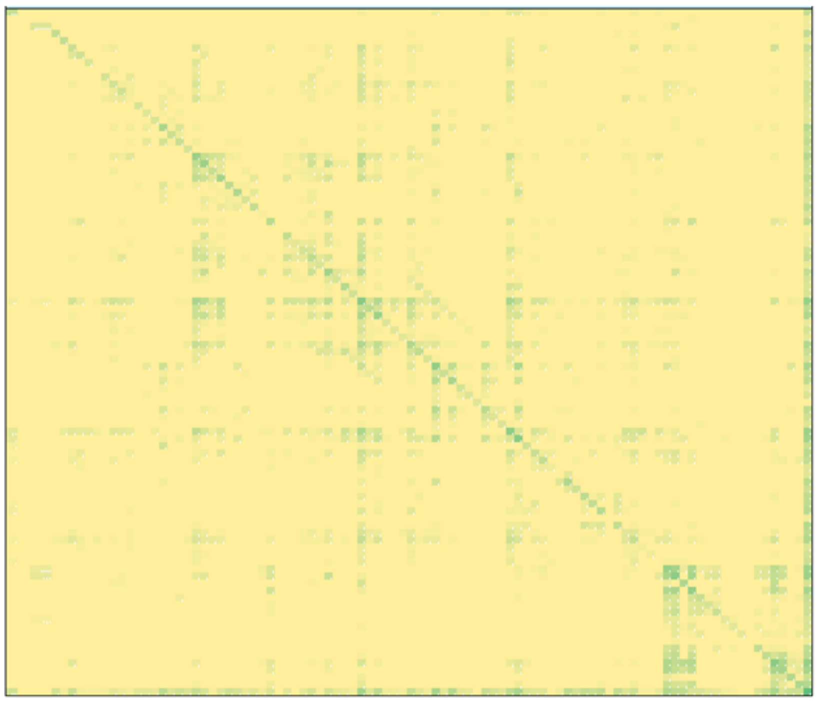


Figure 4.2: Confusion matrix for category detection (in log scale)

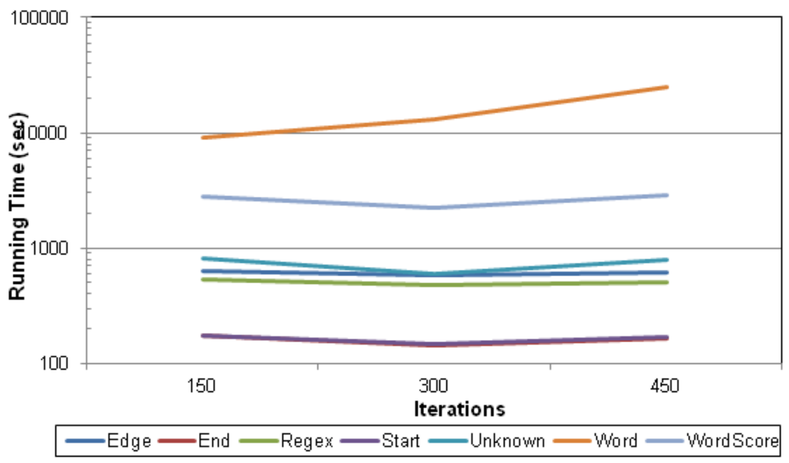


Figure 4.3: Running time w.r.t number of iterations

4.2 Feature Selection

To measure the contribution of each feature to the prediction accuracy, we empirically test different features using two methodologies:

1. *Selecting one feature at a time*: This tests the isolated contribution per feature.
2. *Selecting all features except one at a time*: This tests the drop in accuracy caused by discarding one feature.

The results for these experiments are shown in Table 4.2. According to the results, the most effective features are "Word" and "Wordscore". This means that word identity is in general highly predictive. Since the number of unique words per data set is large, the need for a learner that can handle large feature spaces efficiently is required for an high accurate classifier.

4.3 Combining Class and Boundary Labels

In entity extraction, when a class label and the corresponding boundary label are not compatible, it is not clear which one is correct. For example if the classifier predicts "boundary=begin" and "class=no-entity", it is challenging to decide if the word is the beginning of

Table 4.2: Accuracy per feature class (iteration rate 150)

Features	**Boundary**			**Class**			**Boundary + Class**		
	Prec	**Recall**	**F1**	**Prec**	**Recall**	**F1**	**Prec**	**Recall**	**F1**
ST	36.59%	29.76%	32.82%	35.73%	24.63%	29.16%	36.23%	26.43%	31.57%
ED	36.59%	29.76%	32.82%	35.73%	24.63%	29.16%	37.43%	27.61%	40.98%
EG	38.36%	31.22%	34.43%	36.24%	25.71%	30.08%	39.12%	27.93%	34.49%
UN	67.30%	61.99%	64.53%	65.73%	86.82%	74.82%	70.65%	87.12%	73.15%
WD	72.96%	69.97%	71.43%	69.94%	86.86%	77.49%	72.87%	87.14%	80.06%
WS	72.29%	69.54%	70.89%	69.89%	86.80%	77.43%	71.73%	88.04%	79.34%
RG	35.59%	30.92%	33.09%	32.73%	86.95%	47.56%	36.94%	88.23%	49.65%
PS	63.32%	56.29%	59.59%	60.68%	57.51%	59.05%	64.04%	56.93%	60.27%
LX	69.41%	64.13%	66.67%	66.75%	62.30%	64.45%	70.12%	68.43%	69.26%
ST+ED+EG+UN+WD+WS+PS+LX	93.32%	78.91%	85.51%	80.44%	70.95%	75.40%	82.43%	72.11%	78.16%
ST+ED+EG+UN+WD+RG+PS+LX	93.54%	77.69%	84.88%	80.30%	70.02%	74.81%	81.23%	72.63%	76.31%
ST+ED+EG+UN+WS+RG+PS+LX	87.15%	71.71%	78.68%	74.23%	66.78%	70.30%	75.11%	68.91%	71.23%
ST+ED+EG+WD+WS+RG+PS+LX	92.55%	76.59%	83.82%	79.52%	69.09%	73.94%	82.43%	70.34%	75.32%
ST+ED+UN+WD+WS+RG+PS+LX	69.78%	60.92%	65.05%	68.52%	52.62%	59.52%	72.14%	54.21%	63.29%
ST+EG+UN+WD+WS+RG+PS+LX	93.15%	76.91%	84.26%	80.68%	69.66%	74.77%	83.21%	72.89%	76.01%
ED+EG+UN+WD+WS+RG+PS+LX	93.07%	77.68%	84.68%	80.59%	69.92%	74.88%	81.32 %	71.23%	75.43%
ST+ED+EG+UN+WD+WS+RG+LX	92.43%	76.61%	83.15%	79.40%	68.35%	73.23%	80.14 %	70.64%	74.98%
ST+ED+EG+UN+WD+WS+RG+PS	90.13%	75.54%	82.80%	79.01%	67.64%	72.65%	79.15 %	69.31%	73.56%

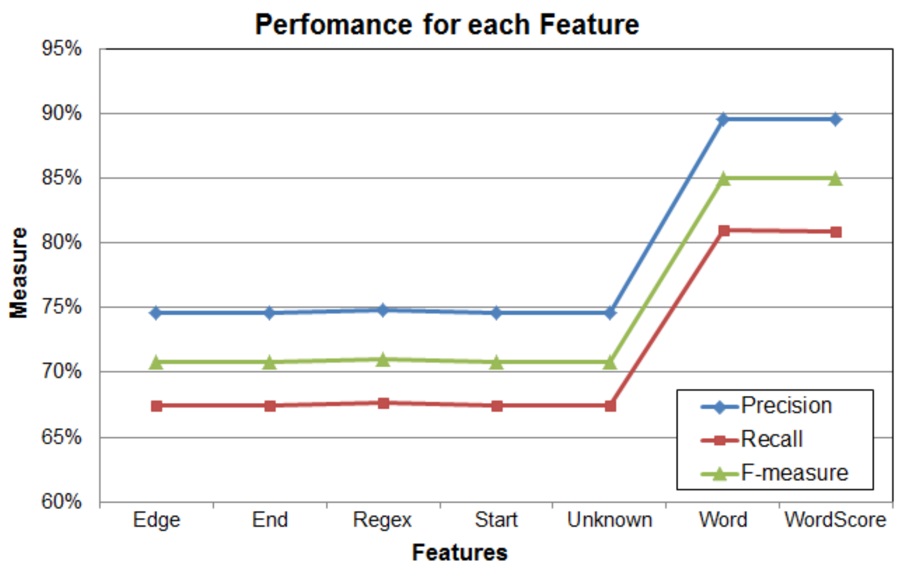


Figure 4.4: Effect of selecting each feature on precision, recall and F-measure.

Table 4.3: Rules and resulting accuracy for boundary and class label combination

Boundary label	Class label	Rule	Accuracy
Begin	No Entity	No Entity → Entity	75.90%
Inside	No Entity	No Entity → Entity	63.00%
End	No Entity	No Entity → Entity	58.30%
Unigram	No Entity	No Entity → Entity	53.80%
Outside	Entity	Entity → No Entity	55.80%

an entity (based on boundary decision) or it is not part of an entity at all (based on class decision). For tackling this problem, we analyzed the occurrence patterns of the incompatible cases and use the training dataset to find correct answers in each case. Table 4.4 summarizes our findings.

In the first five columns of Table 4.4, there are some errors in prediction, but the predicted boundary and class combinations are compatible. In the remaining columns, the combinations are not compatible. For example "BEGIN-NOENTC" it means "boundary=begin" and "class=no-entity". In these cases, we find the most frequent combination of the actual labels for each combination. We use these combinations for inferring rules for incompatible cases as shown in Table 4.3.

It should be noted that, according to the inferred rules in Table 4.3, the best decision

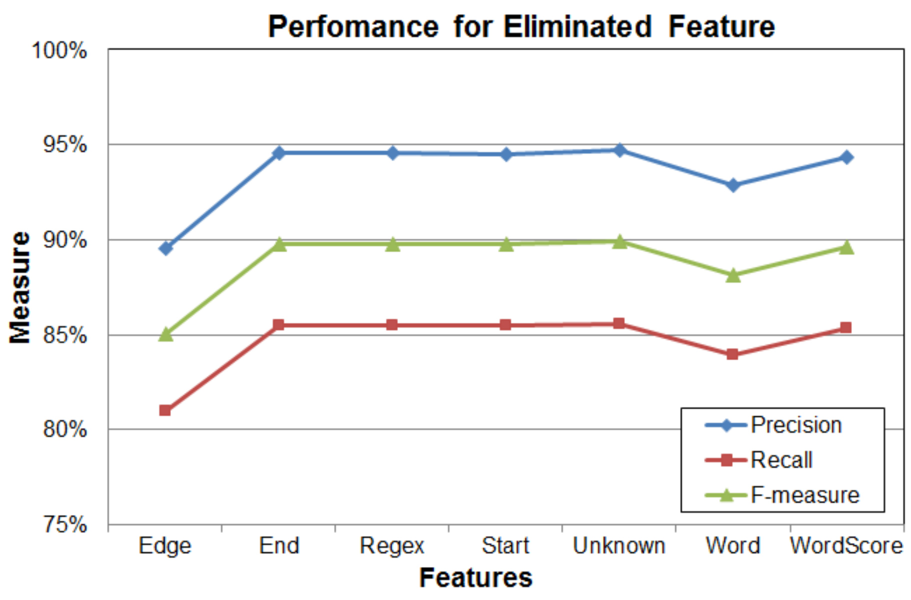


Figure 4.5: Effect of selecting all features except one on precision, recall and F-measure.

is always to change the class label when the boundary label and the class label are not compatible. Thus, we can conclude that the boundary label is more accurate than the class label. However in some cases they are comparable, for example, for "boundary=unigram" and "class=no-entity" the accuracy for relying on the class label is 53.8%, and for going with the boundary label (from unigram to outside) is 45.4%.

4.4 Comparison to Other NER Systems

To compare the proposed classifier with current publicly available packages, we choose two state-of-the-art NER packages: the Illinois Named Entity Tagger [19] and the Stanford Named Entity Recognizer [7].

Illinois Named Entity Tagger[1]: This is a state of the art NER tagger that tags plain text data with named entities. The newest version tags entities with either the "classic" 4-label type set (people, organizations, locations, miscellaneous), while the most recent one can also tag entities with a larger 18-label set (based on the OntoNotes corpus). It uses

[1]http://cogcomp.cs.illinois.edu/page/software_view/NETagger

Table 4.4: Ratio of predicted versus actual boundary and class label combinations.

Boundary-Category	TRUTH				
PREDICTED	unigram-ent	begin-ent	outside-noent	end-ent	inside-ent
unigram-ent	90.30%	1.60%	6.10%	1.80%	0.20%
begin-ent	3.40%	90.30%	1.60%	0.50%	4.20%
outside-noent	1.70%	0.40%	97.30%	0.20%	0.30%
end-ent	3.80%	0.40%	1.90%	91.60%	2.20%
inside-ent	1.30%	6.80%	1.40%	6.70%	83.90%
outside-ent	25.50%	4.50%	**55.00%**	11.90%	3.10%
begin-noent	0.50%	**75.90%**	20.60%	0.00%	2.90%
end-noent	3.20%	0.00%	36.30%	**58.30%**	2.20%
inside-noent	0.60%	6.00%	27.80%	2.60%	**63.00%**
unigram-noent	**53.80%**	0.00%	45.40%	0.80%	0.00%

gazetteers extracted from Wikipedia, word class models derived from unlabeled text, and expressive non-local features. The best performance is 90.8% F1 on the CoNLL03 shared task data. The tagger is robust and has been evaluated on a variety of datasets [19].

Stanford Named Entity Recognizer [2]: Stanford NER is a well-known package for English NER, particularly for common classes (PERSON, ORGANIZATION, LOCATION). Various other models for different languages and circumstances are also available. The software is similar to the baseline local+Viterbi model in [7], but adds new distributional similarity based features. The big models were trained on a mixture of CoNLL, MUC-6, MUC-7 and ACE named entity corpora, and as a result the models are fairly robust across domains.

For our project, we use BBN and CoNLL. Since Stanford NER supports the detection of persons, organizations and locations, we exclude other entities from our model. Actually we use the main type plus specificity model. This model detects additional classes, which we exclude for the assessment. The comparison results are given in Table 4.5.

The complexity of the four class label models in terms of the number of labels correlates with accuracy. The smallest model with 10 classes has the highest overall accuracy (79.8%), and the largest model with 94 classes achieves 75.9%. The boundary prediction (with 5

[2]http://nlp.stanford.edu/software/CRF-NER.shtml

Table 4.5: Comparison: F1 score of the proposed classifier and benchmarks (Stanford-NER and Illinois-NER)

	Stanford-NER	Illinois-NER	All Features
BBN	83.01	86.43	79.82
ACE05	83.56	87.18	77.56

labels) has the highest accuracy (85.6%).

4.5 Parallelization

The performance improvements for the parallelized trainer were measured on a Linux server with 64GB memory and 16 quad core processors. For the boundary detection (5 categories), the benefits of parallelism was are minimal. For class detection, running time decreased by a factor of roughly 10 in the best case. Tests with more input blocks are needed to produce better speed-ups. This performance improvement is available on any multi-core machine.

Chapter 5

Conclusions

In this thesis, we have built an entity recognizer that supports named entities (e.g. persons and organizations) as well as non-named entities (e.g. physical objects). It consists of two sub-modules for predicting the boundary and the class of entities. By using this strategy and combining the result of two classifiers using a rule-based approach, we resolved the data sparseness issue that enabled us to support a large number of entity types. Our model that supports 10 classes (including person, organizations, locations, facilities, objects, events) that entail named and non-named entities and achieves an accuracy of 79.8% (F1).

The proposed solution has many applications in socio-technical systems. It can help researchers to construct network data suitable for answering substantive questions about socio-technical systems.

Our solution has some limitations due to low accuracy for predicting some classes. For the largest model that we constructed (the 94 classes), the accuracy is 75.9%. This is not high in comparison to usual ER packages. In this case, the recall is lower than precision. In future work, we aim to improve recall by using larger datasets for training the boundary and class models. Due to the parallelized learner module, learning on larger datasets from different sources - for having more robust classifier - will be time efficient.

As another direction for future work, we can improve the combining method for category and boundary classifier's results. Our preliminary experiment on the combining method was only scratching the surface of the problem. We believe this kind of technique is the key to improving the NER system's accuracy, without making the system and training phase more complex by adding new features and training datasets.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[2] D. M. Bikel, R. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1-3):211–231, 1999.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM, 2008.

[4] W. W. Cohen and S. Sarawagi. *Exploiting dictionaries in named entity extraction: combining semi-Markov extraction processes and data integration methods.* Paper presented at the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, 2004.

[5] J. Diesner. From texts to networks: Detecting and managing the impact of methodological choices for extracting network data from text data. *Knstliche Intelligenz/ Artificial Intelligence. doi:*, 10., 2013.

[6] T. G. Dietterich. *Machine Learning for Sequential Data: A Review.* Paper presented at the Joint IAPR International Workshops SSPR 2002 and SPR 2002, Windsor, ON, Canada, 2002.

[7] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[8] C. Kirchner and J. W. Mohr. Meanings and relations: An introduction to the study of language, discourse and networks. *Poetics*, 38(6):555–566, 2010.

[9] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[10] K. Leetaru and P. A. Schrodt. *GDELT: Global data on events, location, and tone.* Paper presented at the ISA Annual Convention, 2013.

[11] J. Mayfield, P. McNamee, and C. Piatko. Named entity recognition using hundreds of thousands of features. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 184–187. Association for Computational Linguistics, 2003.

[12] A. McCallum. Information extraction: Distilling structured data from unstructured text. *Queue*, 3(9):48–57, 2005.

[13] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

[14] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[15] J. Milroy and L. Milroy. Linguistic change, social network and speaker innovation. *Journal of Linguistics*, 21:339–384, 1985.

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

[17] P. F. Nardulli, S. L. Althaus, and M. f. Hayes. *A Progressive Supervised Learning Approach to Generating Rich Civil Strife Data*. Sociological methodology, 2015.

[18] T. Poibeau and L. Kosseim. Proper name extraction from non-journalistic texts. *Language and computers*, 37(1):144–157, 2001.

[19] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009.

[20] L. F. Rau. Extracting company names from text. In *Artificial Intelligence Applications, 1991. Proceedings, Seventh IEEE Conference*, volume 1, pages 29–32, Feb 1991.

[21] C. Roth. *Binding social and semantic networks*. Paper presented at the 2nd European Conference on Complex Systems (ECCS) Oxford, UK, 2006.

[22] S. Sarawagi. Information extraction. *Foundations and trends in databases*, 1(3):261–377, 2008.

[23] P. A. Schrodt, . Yilmaz, D. J. Gerner, and D. Hermick. *Coding Sub-State Actors using the CAMEO (Conflict and Mediation Event Observations) Actor Coding Framework*. Paper presented at the Annual Meeting of the International Studies Association, San Francisco, CA, 2008.

[24] S. Sekine and C. Nobata. Definition, dictionaries and tagger for extended named entity hierarchy. In *LREC*, pages 1977–1980, 2004.

[25] S. Sekine and E. Ranchhod. *Named entities: recognition, classification and use*, volume 19. John Benjamins Publishing, 2009.

[26] B. Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107. Association for Computational Linguistics, 2004.

[27] D. Shen, J. Zhang, G. Zhou, J. Su, and C.-L. Tan. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 49–56. Association for Computational Linguistics, 2003.

[28] Y. Tsuruoka and J. Tsujii. Boosting precision and recall of dictionary-based protein name recognition. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 41–48. Association for Computational Linguistics, 2003.

[29] T. Van Holt, J. C. Johnson, K. M. Carley, J. Brinkley, and J. Diesner. Rapid ethnographic assessment for cultural mapping. *Poetics*, 41(4):366–383, 2013.

[30] C. Walker, S. Strassel, J. Medero, and K. Maeda. *ACE 2005 Multilingual Training Corpus*. Linguistic Data Consortium, Philadelphia, 2006.

[31] R. Weischedel and A. Brunstein. BBN pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*, 112, 2005.