Article

JIS

Journal of Information Science
37(2) 137–151

DOI: 10.1177/0165551511398573
jis.sagepub.com
SAGE

Automatic rule learning exploiting morphological features for named entity recognition in Turkish

Serhan Tatar
Department of Computer Engineering, Bilkent University, Turkey

Ilyas Cicekli
Department of Computer Engineering, Hacettepe University, Turkey

Abstract
Named entity recognition (NER) is one of the basic tasks in automatic extraction of information from natural language texts. In this paper, we describe an automatic rule learning method that exploits different features of the input text to identify the named entities located in the natural language texts. Moreover, we explore the use of morphological features for extracting named entities from Turkish texts. We believe that the developed system can also be used for other agglutinative languages. The paper also provides a comprehensive overview of the field by reviewing the NER research literature. We conducted our experiments on the TurkIE dataset, a corpus of articles collected from different Turkish newspapers. Our method achieved an average *F*-score of 91.08% on the dataset. The results of the comparative experiments demonstrate that the developed technique is successfully applicable to the task of automatic NER and exploiting morphological features can significantly improve the NER from Turkish, an agglutinative language.

Keywords
automatic rule learning; morphological features; named entity recognition; Turkish

1. Introduction

Automating the process of finding valuable pieces of information from a staggering amount of data is the main goal of information extraction (IE). IE can be defined as the recognition of selected types of entities, relations, facts or events in natural language texts and the transformation of such recognized information into a structured format. Our research focus is on named entity recognition (NER) from unstructured texts, one of the most fundamental IE tasks, which aims to locate and categorize all instances of predetermined categories of named entities (persons, locations, organizations, etc.) in texts. NER is a prerequisite to more sophisticated information extraction tasks, such as entity relation detection and event extraction.

Domain adaptation, one of the key challenges in the IE field can be described as the process of adapting an extraction system developed for one domain to another. As for the domain itself, it can be thought of as the genre and format of the content in documents from which named entities will be extracted. Adapting knowledge-source-based and rule-based NER approaches to new domains is generally not straightforward since it requires human intervention to first analyse the domain and develop the appropriate resources (i.e. dictionaries, rules) to tackle it. Keeping these resources up-to-date given evolution in domains also requires constant human intervention.

The characteristics of the source language to extract information from also have a significant impact on the extraction techniques being used. A certain feature of one language, which can help the extraction process, may not be available for

Corresponding author:
Ilyas Cicekli, Department of Computer Engineering, Hacettepe University, Ankara, Turkey.
Email: ilyas@cs.hacettepe.edu.tr

another. For example Chinese and Arabic, unlike English, lack the capitalization information which can be used as clues for identifying named entities [1, 2]. Moreover, a language-specific phenomenon can complicate the NER task. For instance, in German, all nouns are capitalized; consequently the number of word forms to be considered as potential named entities is much larger [3]. In Slavonic languages the case of the noun phrase within a numerical phrase depends on the numeral and on the position of the whole numerical phrase in the sentence [4]. Likewise, NER for the languages with complex morphological structures, such as Turkish, requires a morphological level of processing.

In this study, an automatic rule learning method to extract named entities from Turkish news articles is presented. Adopting a supervised learning strategy, the developed NE recognizer automatically starts with a set of named entities collected from a training dataset and generates the extraction rules from the given examples by using a carefully designed learning strategy. Our system employs rule filtering and rule refinement techniques to minimize any possible reduction in accuracy caused by the generalization. In order to obtain accurate generalization, we use several syntactic and semantic features of the text, including: orthographical, contextual, lexical and morphological features. In particular, morphological features of the text are effectively used in this study to increase the extraction performance for Turkish, an agglutinative language that is therefore morphologically rich and productive. The NER-related challenges drawn from the nature of Turkish are explained in more detail in Section 3.1.

The structure of the paper is as follows. Section 2 reviews the previous research in NER. Section 3 describes how we employed automatic rule learning for the task of NER. Section 4 shows the results of the experimental evaluation of the study. Finally, in the last section, we conclude and indicate directions for future research.

2. Brief survey of the previous research

NER has been well-researched and many approaches have been proposed ranging from handcrafted rule-based systems to adaptive learning systems. Numerous studies [5–8] have reviewed the research. Early investigations [9–11] in the IE community established a linguistic architecture based on cascading automata and domain-specific knowledge. The SRI FASTUS system [9] used a series of finite-state transducers that compute the transformation of text from sequences of characters to domain templates and achieved $F = 94\%$ (F-score: harmonic average between accuracy and coverage) on the NER task. The Proteus system [10] also used cascaded finite state transducers to recognize succession events. Grishman [10] reported the overall NER performance of the system as $F = 88.2\%$. The LaSIE-II system [11], developed at the University of Sheffield, used finite state recognition of domain-specific lexical patterns, partial parsing using a restricted context-free grammar and quasi-logical form (QLF) representation of sentence semantics. The performance of the system for the NER task was reported as $F = 89.1\%$. Although these systems have demonstrated remarkable performance, rule development and management is the main issue in these systems. Developing and managing rules by hand requires high human expertise. Constructing IE systems manually has also proven to be expensive [12]. Domain adaptability is also a major issue for these systems since the domain-specific rules constructed in these systems for a domain cannot be easily applied for another domain.

In order to reduce human effort in building or shifting an IE system, significant research in IE has focused on using supervised learning techniques for automated development of IE systems. Instead of having humans create patterns and rules, these models use automatically generated rules via generalization of examples or statistical models derived from the training data. AutoSlog [13], one of the earliest systems, learns a dictionary of patterns, called concept nodes, with an anchor word, most often the head verb, to activate that concept node to extract information from the text. The CRYSTAL system [14] employed inductive learning to construct a concept dictionary from annotated training data. Inspired by inductive logic programming methods, RAPIER [15,16] used bottom-up (specific to general) relational learning to generate symbolic rules for IE. Freitag [17] describes several learning approaches to the IE problem: a rote l15 earner, a term-space learner based on Naive Bayes, an approach using grammatical induction, and a relational rule learner. Freitag also proposed a multi-strategy approach which combines the described learning approaches. Freitag and Kushmerick [18] introduced wrapper induction, identified a family of six wrapper classes, and demonstrated that the wrappers were both relatively expressive, and efficient for extracting information from highly regular documents. Hsu and Dung [19] presented SoftMealy, a wrapper representation formalism based on a finite state transducer and contextual rules. The Boosted Wrapper Induction (BWI) method [20] learns a large number of relatively simple wrapper patterns and combines them using boosting. The Hidden Markov Models (HMMs) are powerful statistical models that have been successfully applied to the task of IE [21–24]. Maximum Entropy Markov Models (MEMMs) [25], Conditional Random Fields (CRMs) [26, 27], Maximum Entropy Models [28], and Support Vector Machines (SVMs) [29, 30] are also used for IE.

The adaptive methods discussed thus far used supervised learning strategy. Supervised methods can quickly learn the most common patterns, but require a large corpus in order to achieve good coverage of the less frequent patterns.

However, annotating a large corpus is not easy. Semi-supervised (or weakly supervised) methods have been developed to overcome the annotated corpus preparation problem. The major technique in this category is called 'bootstrapping'. Bootstrapping methods [31–33] use only a small degree of supervision, such as a set of seeds, at the beginning. A different solution approach to the annotated corpus preparation problem is to mark only the data which can help to improve the overall accuracy. Active learning methods [34–36] try to make this process by selecting suitable candidates for the user to annotate. The methods based on unsupervised learning approaches [37–39] do not need labelled data at all. Shinyama and Sekine [38] successfully obtained rare NEs with 90% accuracy by using the time series distribution of words in news articles.

Although the language subject to most research applications is English, there has increasingly been growing attention to other languages. The shared task of CoNLL-2002 [39] focused on NER for Spanish and Dutch. A year later, German was one of the focus languages in CONLL-2003 [40]. Numerous studies [1, 41, 42] have been conducted on NER in Chinese. Japanese has received a lot of attention as well [43, 44]. Moreover, various studies deal with the development of systems for addressing NER in various languages: Korean [45], French [46], Greek [47, 48], Danish [49], Italian [33], Vietnamese [50], Bengali [51], Arabic [2], Bulgarian [52], Russian [53], and Ukrainian [54]. Multilingual NER has also received a lot of attention [54, 55]. Cucerzan and Yarowsky [56] presented a language-independent bootstrapping algorithm and conducted their experiments on several languages: Romanian, English, Greek, Turkish and Hindi. Their algorithm, tested on a relatively small corpus, achieved levels of F = 53.04% for NER from Turkish texts. Tur et al. [57] applied statistical learning approaches to a number of tasks for Turkish: sentence segmentation, topic segmentation and name tagging. Their named tagging approach is based on n-gram language models embedded in hidden Markov models. Combining four different in-formation sources (lexical, contextual, morphological and tag sequence), their name tagger reached an F-measure of 91.56%. Bayraktar and Temizel [58] reported F = 81.97% on a corpus of 200 news articles in Turkish using local grammar approach. Kucuk and Yazici [59] presented a rule-based NER system for Turkish which employs a set of lexical resources and pattern bases for the extraction of named entities. Conducting their experimentation on different text genres (news articles, historical text and children's stories), the best performance value they reported is F = 78.7% for the task of NER from Turkish news articles.

In conducting our literature survey we have come to the conclusion that much of the research carried out in NER has been devoted to only a small number of languages. Few papers have been published in relation to Turkish. In this paper, we present an automatic rule learning algorithm that can identify the named entities located in Turkish texts. Our method utilizes a supervised learning strategy and does not rely on handcrafted rules/patterns. Therefore, the system does not suffer from domain adaptability problems. Besides a novel rule learning algorithm, it uses several generalization features and an expressive rule representation language to obtain accurate generalization and remedy the issues related to the data sparseness problem. Moreover, the presented system tries to address the difficulties inherent in the agglutinative languages by exploiting morphological features.

3. Method

3.1. Turkish morphology and named entity recognition

Turkish is a member of the Oghuz group of the Turkic languages, which belongs to the Altaic branch of Ural-Altaic language family. It uses a Latin alphabet consisting of 29 letters, of which eight are vowels and 21 are consonants. Similar to Hungarian and Finnish, Turkish has vowel harmony and lacks grammatical gender distinction.

Another major feature of Turkish is that it is an agglutinative language with free-word order [60]. Although this nature of the language does not have a significant role to play in the NER task, the complex morphological structure of Turkish words makes the task even more difficult. In Turkish, a sequence of inflectional and derivational morphemes can be added to a word. This concatenation process can yield relatively long words, which can convey the equivalent meaning of a phrase, or even a whole sentence in English. A single Turkish word can give rise to a very large number of variants, which results in the vocabulary explosion. Table 1 lists several formations produced using the stem word İstanbul. Note that the morphemes added to the stem word produce different surface forms. The list can easily be expanded (e.g. İstanbul'dakilerdenmiş, İstanbul'dakilerdenmişce,…). In fact, millions of different surface forms can be derived from a nominal or verbal stem [61]. Although, in English, it is possible that a suffix can change the surface form of a proper noun (e.g. Richard's, or IBM's), it is not as common as in Turkish and other morphologically rich languages. Using each surface form generated from the same stem as a different training element would cause data sparseness problems in the training data, which indicates that morphological level processing is a requirement for Turkish NER.

Table 1. Several surface forms produced using the stem word İstanbul

Surface form	Morphological decomposition	English meaning
İstanbul	istanbul +Noun +Prop +A3sg +Pnon +Nom	Istanbul
İstanbul'da	istanbul +Noun +Prop +A3sg +Pnon +Loc	in Istanbul
İstanbul'daki	istanbul +Noun +Prop +A3sg +Pnon +Loc^DB+Adj+Rel	the (one) in Istanbul
İstanbul'dakiler	istanbul +Noun +Prop +A3sg +Pnon +Loc^DB+Adj+Rel^DB+Noun+Zero + A3pl +Pnon +Nom	the ones in Istanbul
İstanbul'dakilerden	istanbul +Noun +Prop +A3sg +Pnon +Loc^DB+Adj+Rel^DB+Noun+Zero + A3pl +Pnon +Abl	from the ones in Istanbul

+*Noun* => Noun; +Prop => Proper Noun ; +Pnon => Pronoun (no overt agreement); +A3sg => 3rd person singular; +A3pl => 3rd person plural; +Nom => Nominative; +Loc => Locative; +Abl => Ablative; ^DB+Adj+Rel => Derivation Boundary + Adjective + Relative; ^DB+Noun+Zero => Derivation Boundary +Noun + 0 Morpheme;

3.2. Generalization features

Two important criteria that determine the efficacy and the success of an NE recognizer are:

- the ability to recognize unseen named entities;
- the ability to precisely distinguish name entities that belong to an NE class from the other NE classes and non-entity names.

Both criteria require accurate generalization of the known named entities. Generalizing means to recognize the parts susceptible to being changed in new names, and represent them with generic placeholders. In our study, we generalize named entities by using a set of features that are capable of describing various properties of the text. In addition to accurate generalization, the use of generalization features will help overcome the data sparseness problem that occurs because of the diversity of the language constructs and the insufficiency of the input data. When we consider all possible language constructs, it is not possible to observe most of the sequences during the training of the language model. The features used in our study can be grouped into the following four categories:

- *Lexical features:* NER deals with text documents which can be seen as contiguous series of tokens. As basic constituents of the text, the tokens themselves are used for NER as well as the features associated with them. Gazetteer information (e.g. list of person names, list of city names) provided to the system can be mapped to the tokens and utilized for generalization purposes. We used a two-level gazetteer hierarchy in our study to achieve accuracy in generalization. The first level in our hierarchy corresponds to each NE class (e.g. *Person*, *Location* etc.) and provides a higher level of generalization. The second level details the gazetteer categorization (e.g. *Location.Country*, *Location.City* etc.) and provides more specific classification.
- *Morphological features:* we effectively used the morphological features of the tokens not only for addressing the challenges arising from the agglutinative nature of Turkish, but also for the clues they offer towards better generalization.
- *Contextual features:* the information captured in the surrounding text of the named entities is used by the system to learn and represent the patterns and regularities in the target-NE boundaries which exist in the training dataset.
- *Orthographic Features:* these features express various orthographic aspects of the tokens. We selected four primitive features, a combination of which can yield more complicated patterns: *Capitalization (Lower, Upper, Proper, Unclassified)*, *Length Class (Short, Middle, Long)*, *Length (the length of the token)*, and *Type Class (Alpha, Numeric, Alphanumeric, Punctuation, Unclassified)*.

3.3. Rule representation and expressiveness

The ability to recognize the regularities among the target-named entities and to capture the learnt regularities/patterns in the rules requires a powerful rule representation. Moreover, the coverage of the learnt rules, which is also related to rule representation, affects the performance of the rule learning systems. While over-specific rules may cause low recall, over-general rules cause low precision. An expressive rule language that can handle the mentioned generalization features in a flexible manner and provide the adequate level of granularity for rule coverage is necessary to achieve good NER performance.

An extraction rule defined in our system consists of four parts. The first simply addresses the NE class which is the target of this rule. The last three contain the pattern segments:

- the *PRE-FILLER (PRE)* segment tries to sense and match the text immediately preceding the target NE;
- the *FILLER (FILL)* segment tries to sense and match the target NE;
- the *POST-FILLER (POST)* segment tries to sense and match the text immediately following the target NE.

The pattern segments in an extraction rule can be seen as a sequence of pattern elements whose type can be either *Similarity (SIM)* or *Optional (OPT)*. A type *SIM* pattern element matches exactly one token from the document that meets the element's constraints. On the other hand, a type *OPT* pattern element can match either a token that meets the element's constraints or none. Being tailored parallel to the features mentioned in the previous section, our rule representation uses pattern elements containing several fields to provide the expressiveness we need for accurate generalization.

In order to make the rule representation concept clearer, two example rules are provided in Figure 1. Rule parts are separated by colons. As the first part of the rule indicates, the first rule captures the pattern information belonging to *PERSON* NE class. The next part, the *PRE* segment, in the first example contains only one pattern element, a *SIM* element, which has eight fields separated by semicolons:

- token;
- morphological tag;
- low-level gazetteer set;
- high-level gazetteer set;
- case tag;
- length class;
- token length;
- type class.

Each field in the pattern element represents a constraint. In order for a pattern element to match a token in the text, that token must satisfy all of the constraints imposed by that pattern element *(conjunction of the constraints)*. The first constraint, token field, is a non-case-sensitive atomic field that matches only the exact text values, if it is set to a value. The morphological tag stored in the second field is a simple regular expression that provides a concise and flexible means for matching tokens according to their morphological characteristics. Each morpheme in the morphological tag is separated by a plus sign. Special characters are used to express the specific variations in the regular expressions. The star (*) character matches a single morpheme with any type. The question mark (?) character with the combination of parentheses indicates that there is zero or one of a single morpheme with the morpheme type given in the parentheses. When it is used with the star character (i.e. '?(*)'), it indicates that there is zero or one of a single morpheme with any type. The third and the fourth field contain gazetteer set constraints. To be matched, a token must be in the coverage of the gazetteer sets, and it must not be included in any gazetteer list which is not in the sets. Since it is practically impossible to have complete gazetteer lists that cover all available names, our matching algorithm does not refuse the tokens which are not listed in any of our gazetteer lists in order to enable the system to tolerate the potential faults that can be caused by missing names in the gazetteer lists. The remaining atomic fields contain the constraints regarding the orthographic features described in the preceding section. Returning to the first example given in Figure 1, the *PRE* segment of the rule indicates that the NE phrase to be matched by this rule must be preceded by the token *Valisi* (Governor (of)).

[PERSON :

SIM<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha***> :***

SIM< ; *+Noun+?(Prop)+A3sg+Pnon+Nom; {Person.First_Name}; {Person}; Proper; ; ; Alpha>
OPT< ; *+Noun+?(Prop)+A3sg+ Pnon+Nom; {Person.First_Name}; {Person}; Proper; ; ; Alpha>
SIM< ; *+?(Noun)+*+*+*+*; {Person.Last_Name}; {Person}; Proper; ; ; Alpha***> :***

SIM<,; ,+Punc; {}; {}; Unclass; Short; 1; Punc***> /***

[DATE :

NULL :

SIM< ; *+Num+Card; {Date.DayNumber, Time.Minute, Number.Number}; {Date, Time, Number}; Unclass; Short; 2; Number>
SIM< ; *+Noun+A3sg+Pnon+Nom; {Date.Month}; {Date}; ; ; ; Alpha>
OPT<,; ,+Punc; {}; {}; Unclass; Short; 1; Punc>
SIM<; *+Num+Card; {Date.Year}; {Date}; Unclass; Short; 4; Number> ***:***

SIM< ; *+Noun+A3sg+P3sg+*; {Date.PostPhrase}; {Date}; Lower; ; ; Alpha***> /***

Figure 1. Example rules.

The *FILL* segment states that the NE phrase to be matched by this rule must start with a token which is a name in nominative form, not listed in any gazetteer list other than *Person.First_Name* gazetteer list, in proper case, in any length and containing only alpha characters. The rule also requires that the last token of the person phrase must be either in *Person.Last_Name* gazetteer list or none, in proper case, in any length and containing only alpha characters. Optionally, another token possessing the same characteristics as the first token can occur between the first and the last token. Finally, the *POST* segment asserts that the NE phrase to be matched by the rule must be followed by a comma. The rule given in the second example describes a pattern belonging to the *DATE* NE class in a similar fashion. Note that the *PRE* segment is set to *NULL* value, which means that the rule does not impose any constraints on the tokens that can occur before the target NE phrase. Some text excerpts containing named entities that match the given example rules are shown in Figure 2.

Example Rule 1 (Person):
…Adana Valisi *Cahit Kıraç*, Türk-Amerikan Derneği binasındaki patlamanın konulan bombadan…
…İstanbul Valisi *Muammer Güler*, Çapa'da İETT otobüsünde…
…inceleme yapan Ağrı Valisi *Halil İbrahim Akpınar*, yangının terör…
Example Rule 2 (Date):
…Van'da *31 ekim 2005* tarihinde Erek Polis Karakolu'na…
…*20 Mayıs 200*3 tarihinde Ankara Kızılay'da bir kafede…
…göre, *26 Eylül 2000* günü akşam saatlerinde…

Figure 2. Text excerpts containing named entities that match the example rules given in Figure 1.

3.4. Automatic rule learning

The ability of rule learning and subsequent generalization is one of the critical functions in the system. Our automatic rule learning and generalization method is based on the concept of specific generalization of strings as described in [62]. We applied the concept to generalize both the patterns and the features in different levels and employed the coverage algorithm presented in the study for inducing the rules. The idea behind the concept is to generalize the strings by processing similarities (substrings occurring in both strings) and differences (substrings differing between strings) between them. In order to generalize two strings, a unique match sequence of those strings is obtained, and the differences in the unique match sequence are replaced by variables to get a generalized form of the strings. A unique match sequence can be described as a sequence of similarities (substrings occurring in both strings) and differences (substrings differing between strings) between two strings. Referring the reader to [62] for the details, we will focus more on how we adopted the concept to automatic rule learning for NER.

Prior to learning NER rules from the examples in the training text, the input text is segmented in sentences and tokenized. We followed the standard tokenization method using white-space and punctuation characters as delimiters except that we removed the apostrophe symbol (‘) from our delimiter set since it is meaningful during morphological disambiguation. The next step is to assign feature values to the tokens in the text. First, possible morphological parses of each observed token are found, and the most appropriate one among them is selected through morphological analysis [63] and morphological disambiguation processes [64]. The used morphological disambiguation tool uses a hybrid technique which combines statistical information, hand-crafted grammar rules and transformation-based learning. The reported [65] accuracy morphological disambiguation process on the average is 93.5% when whole morphological parses are considered in calculation. Then, the system continues with a gazetteer list search. An important point to highlight is that the stem of the token is looked up in the gazetteer lists to minimize the effect of morphological complexity. Finally, each token is labelled with their corresponding orthographic feature values before the rule learning. Thus each token is represented by its eight features (token, morphological tag, low-level gazetteer set, high-level gazetteer set, case tag, length class, token length and type class), and NER rules are learnt from them.

Subsequent to pre-processing the training data, the learner starts generating simple patterns from the training examples. A simple pattern is generated using the preceding token, the following token, and the tokens of an NE. The generated simple patterns are kept in the rule representation stated in the previous section. The generalization function $GEN(R_1, R_2)$

takes two examples in the rule representation and returns a generalized rule that covers both examples by relaxing the corresponding constraints specified by the feature fields in the pattern elements. The generalization of the atomic fields (token, capitalization, length class, length, type class) is straightforward. If the values in the corresponding fields of the examples are the same, this value is assigned to the same field of the generalized pattern element; otherwise, the field is left empty, which indicates that no constraint is defined for that field. Generalization of a morphological tag field is based on the concept of a specific generalization of strings [62]. Processing similarities and differences between the morphological tag fields of the examples, and replacing the discovered differences with either a type *ANY* (*) variable or a type *OPTIONAL (?)* variable, or combination of these two, the function obtains a simple regular expression that can represent both examples. For gazetteer set fields, the generalization operation returns the union of two input gazetteer sets. However, if one of the gazetteer sets is empty, the generalization operation returns an empty set. Empty gazetteer sets impose no constraint on the patterns.

Because the examples can differ in the number of tokens, we applied a method that calculates a simple similarity score for each possible way for matching the tokens and selects the match with the maximum similarity score. The similarity score for a match is the sum of the similarity scores of each pattern element, which is the aggregated sum of the similarity scores of each feature field. Each field's contribution to the similarity score varies according to its discrimination power. The field contributions were determined through experimentation. For instance, the orthographic type class is apparently less powerful than gazetteer list sets in discriminating NEs.

In order to illustrate the rule generalization concept, an example rule generation is given in Figure 3. In the example, a generalized rule is learnt from two person name instances located in the following text excerpts: '…Elazığ Valisi *Kadir Koçdemir'in* geçtiği…', '…Van Valisi *Hikmet Tan*, konvoyuna …'. By performing several generalization operations over the different pattern elements, the learner obtains the generalized rule shown in Figure 3. In the example, the generalized rule asserts the following constraints:

- the NE phrase to be matched by this rule must be preceded by the token *Valisi*;
- the NE phrase to be matched by this rule must start with a token which is a name in nominative form, listed in Person.First_Name and/or Person.Last_Name gazetteer lists, in proper case, five to eight characters in length and containing only alpha characters
- the last token of the NE must be in Person.First_Name and/or Person.Last_Name gazetteer lists, in proper case, in any length and containing only alpha characters
- the token that comes after the NE must match the morphological tag specified by the *POST* segment of the rule.

Note that some person names recognizable by the generalized rule are also given in the example. As evident from the given list, exploiting morphological features increases the recognizability of the NEs.

Our coverage algorithm that finds *RULES*, the set of generalized rules, is given in Figure 4. Initially, *RULES* is set to an empty set (Figure 4, Line 1). The algorithm then generates a simple pattern rule *R* for each positive example available in the training dataset and adds *R* into *RULES* if it is not already in the set (Figure 4, lines 2–4). Afterwards, the algorithm iteratively generalizes the rules available in *RULES* (Figure 4, lines 5–19). In each iteration, for each rule R_1 in *RULES*, possible generalized rules that are obtainable by the generalization of that rule with another rule are found and kept in a temporary rule-set $RULES_{temp}$ (Figure 4, lines 9–12). Then, the rules in $RULES_{temp}$ are sorted in descending order of the similarity factor, a score that is directly proportional to the similarity of the rules used to generate a generalized rule (Figure 4, line 13). The sort process is performed to ensure that the rules with high similarity factors are added into *RULES* in the next step. Subsequently, until *k* number of generalized rules are added into *RULES* or every rule in $RULES_{temp}$ are validated, the rules in $RULES_{temp}$ are validated on the training dataset in order to give their confidence factors; and the rules with confidence factors above a certain threshold value are added into *RULES*, while the rules from which the generalized rule are generated are dropped (Figure 4, lines 14–19). The confidence factor of a rule is calculated as the percentage of correctly extracted names (number of correctly extracted/number of extracted) as a result of applying that rule to the training dataset. During the confidence factor calculation, the algorithm also collects the rule exceptions and builds the rule exception list for each rule, which we will discuss in the next section. This iterative loop continues until no more generalized rules with confidence factors above the threshold value can be added into *RULES*. After sorting the *RULES* in ascending order of the coverage – number of positive examples covered – (Figure 4, line 10), the algorithm eliminates the redundancy in *RULES*. If all positive examples covered by a rule are also covered by some other rules in the rule-set, that rule is deleted from the set (Figure 4, lines 11–15). The reasoning behind sorting rules in ascending order of their coverage is our preference of general rules to specific rules.[1]

Seed Instances (Person):

"…Valisi *Kadir Koçdemir'in* geçtiği…", "…Valisi *Hikmet Tan*,…"

Preprocessing:

<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha>
<kadir; kadir+Noun+Prop+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Middle; 5; Alpha>
<koçdemir'in; koçdemir+Noun+Prop+A3sg+Pnon+Nom; {Person.Last_Name}; {Person}; Proper; Long; 11; Alpha>
<geçtiği; geç+Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom; { }; { }; Lower; Middle; 7; Alpha>

<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha>
<hikmet; hikmet+Noun+A3sg+Pnon+Nom; {Person.First_Name}; {Person}; Proper; Middle; 6; Alpha>
<tan; tan+Noun+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Short; 3; Alpha>
<,; ,+Punc; { }; { }; Unclass; Short; 1; Punc>

Simple patterns:
[PERSON:

SIM<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha> ***:***

SIM<kadir; kadir+Noun+Prop+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Middle; 5; Alpha>
SIM<koçdemir'in; koçdemir+Noun+Prop+A3sg+Pnon+Nom; {Person.Last_Name}; {Person}; Proper; Long; 11; Alpha>***:***

SIM <geçtiği; geç+Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+P3sg+Nom; { }; { }; Lower; Middle; 7; Alpha> ***/***

[PERSON:

SIM<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha>***:***

SIM<hikmet; hikmet+Noun+A3sg+Pnon+Nom; {Person.First_Name}; {Person}; Proper; Middle; 6; Alpha>
SIM<tan; tan+Noun+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Short; 3; Alpha>

SIM<,; ,+Punc; { }; { }; Unclass; Short; 1; Punc>

Generalized rule:

[PERSON :

SIM<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha> ***:***

SIM< ; *+Noun+?(Prop)+A3sg+Pnon+Nom; { Person.First_Name, Person.Last_Name}; {Person}; Proper; Middle; ;Alpha>
SIM< ; *+Noun+?(Prop)+A3sg+Pnon+*;{ Person.First_Name, Person.Last_Name}; {Person};Proper; ; ;Alpha> ***:***

SIM< ;*+?(Verb)+?(Pos)+?(^DB)+?(Adj)+?(PastPart)+?(^DB)+?(Noun)+?(Zero)+?(A3sg)+?(P3sg)+*; { }; { }; ; ; ; > ***/***

Recognizable NEs:

"Adana Valisi *Cahit Kıraç*,", "Tunceli Valisi *Mustafa Erkal* yaptığı", "Çankırı Valisi *Ayhan Çevik'in* bulunduğu…", "İstanbul Valisi *Muammer Güler*,"

Figure 3. An example rule generation.

There are two parameters determined by the user: the confidence factor threshold (T), and the number of generalized rules to generate for each rule in each cycle (k). The first parameter controls the trade-off between selectivity and sensitivity. By setting a higher T value, it is possible to increase the system's ability to precisely distinguish name entities that belong to an NE class from the other NE classes and non-entity names; however, this can result in a decrease in the system's ability to recognize unseen named entities. The second parameter controls the learning time of the system. The confidence factor of a rule is found by validating that rule on the training dataset, which requires a computational time. By limiting the number of generalized rules to generate for each rule in each cycle, the algorithm provides control over the total computational time spent for confidence factor calculation. This parameter becomes particularly important for large datasets, since the algorithm suggests a large initial space of rule candidates.

In order to make full use of the information available in training data and improve the algorithm's extraction performance by further rule refinement, each rule in the rule-set is associated with a set of exceptions. The problem is that of efficient utilization of the negative examples (i.e. non-NEs or NEs of different classes) in the training data, though they are used in confidence factor calculations (Figure 4, line 15). Unless it is a 100% confident rule, a rule in the final rule-set may cover some negative instances in the training data. This leads to recognition of an incorrect NE during the test, even if that name is marked as a non-NE or an NE of a different class in the training data. This issue is solved by associating each rule in the final rule-set with a set of exceptions. During confidence factor calculation, every negative instance recognized by the candidate rule is put into that rule's exception set. If any of the names in a rule's exception set are recognized by that rule during the test, the recognized names are just ignored and not extracted.

```
(1)  RULES ← { }.
(2)  For each positive example e in the example space E:
(3)      R ← e.
(4)      If R ∉ RULES Add R into RULES.
(5)  hasMoreGeneralization ← TRUE
(6)  while (hasMoreGeneralization = TRUE)
(7)      hasMoreGeneralization ← FALSE
(8)      For each rule R1 in RULES:
(9)          RULES_temp ← { }.
(10)         For each rule R2 in RULES (where R1 ≠R2):
(11)             R ← GEN(R1, R2).
(12)             If R ∉ RULES_temp Add R into RULES_temp.
(13)         Sort RULES_temp in descending order of the similarity factor
(14)         Until k rules added into RULES or every rule in RULES_temp validated:
(15)             Test every rule R in RULES_temp on the training dataset and Calculate CF_R (the confidence factor of R).
(16)             If CF_R >= T (confidence factor threshold) and R ∉ RULES
(17)                 Add R into RULES.
(18)                 Drop rules R1 and R2 ,from which R generalized, from RULES
(19)                 hasMoreGeneralization ← TRUE
(20) Sort RULES in ascending order of the coverage
(21) For each rule R in RULES:
(22)     If there exists another rule that covers E_R (the positive examples covered by R)
(23)         Drop R from RULES
(24)     If every example in E_R is also covered by another rule in RULES
(25)         Drop R from RULES
```

Figure 4. The rule generalization algorithm.

3.5. Testing and post-processing

Subsequent to the generation of the rule-set and the completion of the training phase, the test phase starts. There is no specific order to apply the learnt rules. We consider all of the rules, with a confidence factor over the threshold value, as valid rules. Therefore, starting from each token in the text, the system applies the learnt rules to the test data. If a rule matches a phrase between two sentence boundaries and the matched phrase is not in that rule's exception set, the matched phrase is put into a candidate list. In case of any conflict (i.e. overlapping phrases), the longest candidate is selected during the post-processing step, which comes after testing.

4. Experimental evaluation

4.1. Data and methodology

We conducted a set of experiments in order to evaluate the performance and the behaviour of the proposed NER method under different conditions. The main objective of the experimentation is to analyse the performance and behaviour of our method on realistic data, with different setups. We also compared our methods to several other studies. Moreover, we investigate the impact of using morphological features and the other novelty we proposed on the extraction process.

A major obstacle to Turkish NER is the scarcity of publicly available annotated corpora. We conducted our experiments on the TurkIE[2] corpus. In order to generate the corpus, we manually tagged 355 news articles on terrorism from both online and print news sources in Turkish. Tagging was performed using an IE-tagging tool,[3] which is developed to allow the users to tag texts using a graphical interface in a user-friendly environment. Before annotating the named entities, the selected articles were tokenized, split into sentences and segmented into topics. The TurkIE corpus contains 54,518 tokens, 3524 sentences and 545 topics. We followed the MUC-7 NE task definition [66] as a guideline for annotation. 5672 NEs were tagged in five categories: 1335 person names, 2355 location names, 1218 organization names, 373 date expressions and 391 time expressions. In order to evaluate the developed method, we performed 10-fold cross validation on the data-set, and the average number of the learned rules above the threshold value was approximately 510.

We measured precision, recall, and *F*-score, as is commonly done in the Message Understanding Conference (MUC) evaluations. Precision is the percentage of extracted named entities that are correct. Recall can be defined as the fraction of correct outcomes divided by the total number of possible correct answers. The *F*-score, harmonic mean of precision and recall, provides a method for combining precision and recall scores into a single value. We used the exact criteria which

is the most conservative approach to determine the truth value of the matching. In our experiments, a predicted instance is not considered as a correct match unless it matches exactly an actual instance in the text. Moreover, in our scoring, the expectation from the system is to find all occurrences of the named entities.

4.2. Results and discussion

4.2.1. Quantitative results and comparison of the methods. Table 2 shows the quantitative results of the experiments performed. Our automatic rule learning method achieved overall performance of $F = 91.08\%$ on the dataset. The developed system reached best performance score $F = 96.5\%$ on locating *DATE* fields; the system extracted 97.69% of the *DATE* fields in the test dataset and 95.34% of the found *DATE* fields were correct. It achieved $F = 87.68\%$ on a more challenging NE type, *ORGANIZATION*.

The system produced better results than many of the previous studies. One important point to highlight before discussing the results of the comparisons is that they were not made on the same datasets, since the datasets were not publicly available. Conducting their experiments on a relatively small corpus, Cucerzan and Yarowsky [56] reported $F = 53.04\%$ for NER from Turkish texts. They aimed at building a maximally language-independent system for NE recognition and classification, which lead using minimal information about the source language. Following unsupervised learning strategy, they used an EM-style bootstrapping algorithm based on iterative learning and re-estimation of contextual and morphological patterns captured in hierarchically smoothed trie models. Our method uses supervised learning. With a focus on only Turkish, our algorithm is designed to take advantage of the specific characteristics of the Turkish language and to address any challenges pertaining to it. Moreover, our algorithm is capable of utilizing several resources (e.g. dictionaries and gazetteer lists) to obtain better extraction performance. It should be noted that unsupervised methods are advantageous over supervised methods in terms of domain adaptation, since they do not need training data.

Bayraktar and Temizel [58] reported $F = 81.97\%$ for person name extraction from Turkish text using a local grammar approach. They focused on reporting verbs (e.g. said, told) in the sentences, and used these reporting verbs as the clues for extracting person names. Their study covered finding significant reporting verbs in Turkish and obtaining hand-crafted extraction patterns by conducting concordance analysis by using the found forms of reporting verbs. The rule-based system developed by Kucuk and Yazici [59] achieved $F = 78.7\%$ for the task of NER from Turkish news articles. Their system heavily relies on a manually compiled set of lexical resources (e.g. dictionary of person names, list of well-known people) and hand-crafted pattern bases to extract each NE type in its scope. In order to mitigate the issues caused by the agglutinative nature of Turkish, they used a morphological analyser. The system is designed to extract the phrases which exist in the provided lexical resources, conform to the patterns in the pattern bases or the inflectional forms. Both systems [58, 59] depend heavily on the lexical resources and the manually developed rules/patterns. One drawback is the high human expertise required for the development and management of the rules. Another well-known shortcoming is their adaptability to new domains. For instance, Kucuk and Yazici [59] reported that their system scored $F = 69.3\%$ and $F = 55.3\%$ on *Child Stories* and *Historical Text* domains respectively.

The statistical learning system presented by Tur et al. [57] reached $F = 91.56\%$. When the learning strategy and the information sources used are considered, their system is the most similar previous work. Both our approach and theirs use supervised learning strategy. Furthermore, both methods exploit similar features (lexical, morphological, contextual, etc.), though there are differences in the way the features are utilized. They used hidden Markov models for NER, and we employ automatic rule learning approach for the same task. Their system only determines person, location and organization as NEs, but our system also recognizes time and date entities.

Table 2. Quantitative performance results of the system

NE category	Precision (%)	Recall (%)	*F*-score (%)
Person	96.69	92.08	94.33
Location	90.20	89.86	90.03
Organization	87.36	88.01	87.68
Date	97.69	95.34	96.50
Time	93.12	91.00	92.05
Overall	91.74	90.43	91.08

The last row shows the overall extraction performance of the developed system. We use the standard formula for precision, recall and *F*-score calculation: precision = (true positives)/(true positives + false positives); recall = (true positives)/(true positives + false negatives); *F*-score = (2 * precision * recall)/(precision + recall).

4.2.2. Error analysis. The analysis conducted revealed that the extraction errors more frequently occur in the following cases:

- *Nested NEs*: nested NE constructs are common forms observed in texts. Following MUC-7 NE task definition [65], only the longest NE was tagged in case of nested NEs. For instance, *'5 Ocak Caddesi' (5th January Street)* is tagged as one location name: <*LOCATION*>*5 Ocak Caddesi*</*LOCATION*>, instead of a date and a location name: <*LOCATION*><*DATE*>*5 Ocak*</*DATE*> *Caddesi*</*LOCATION*>. Recognizing the inner NE (e.g. *'5 Ocak'*), and missing the outer one (e.g. *'5 Ocak Caddesi'*) is observed as one type of erroneous extraction.
- *Long NEs*: partial detection of the long NEs, especially long *ORGANIZATION* and *LOCATION* names, is another frequent type of erroneous extraction. The analysis showed that the average system performance for the long NEs is below the overall performance of the system.

4.2.3. Threshold factor. The coverage algorithm has a user-set threshold parameter which has an impact on the performance of our extraction method. Figure 5 shows the performance of the automatic rule learning method as the threshold parameter changes. The optimum value for the threshold parameter is found to be 0.87, where the acquired *F*-score value is maximized, through experimentation.

In the first half of the graph, we observe a continuous climbing trend in the precision and recall at the same time. The increase in the precision parallel to the increase in the threshold value is a normal behaviour. However, one would expect inversely proportional relation between the recall and the threshold. The observed situation is due to the fact that the longest candidate among the conflicting candidate phrases is selected during the post-processing step. In the second half, as expected, the recall value decreases and the precision value increases with the increase in threshold. Another notable observation is the local drops in the recall rate where the threshold parameter is 0.5. This behaviour is caused by the elimination of a general rule whose true positive (TP) returns are more than its false positive (FP) returns. A similar situation occurs where the threshold parameter is 0.8.

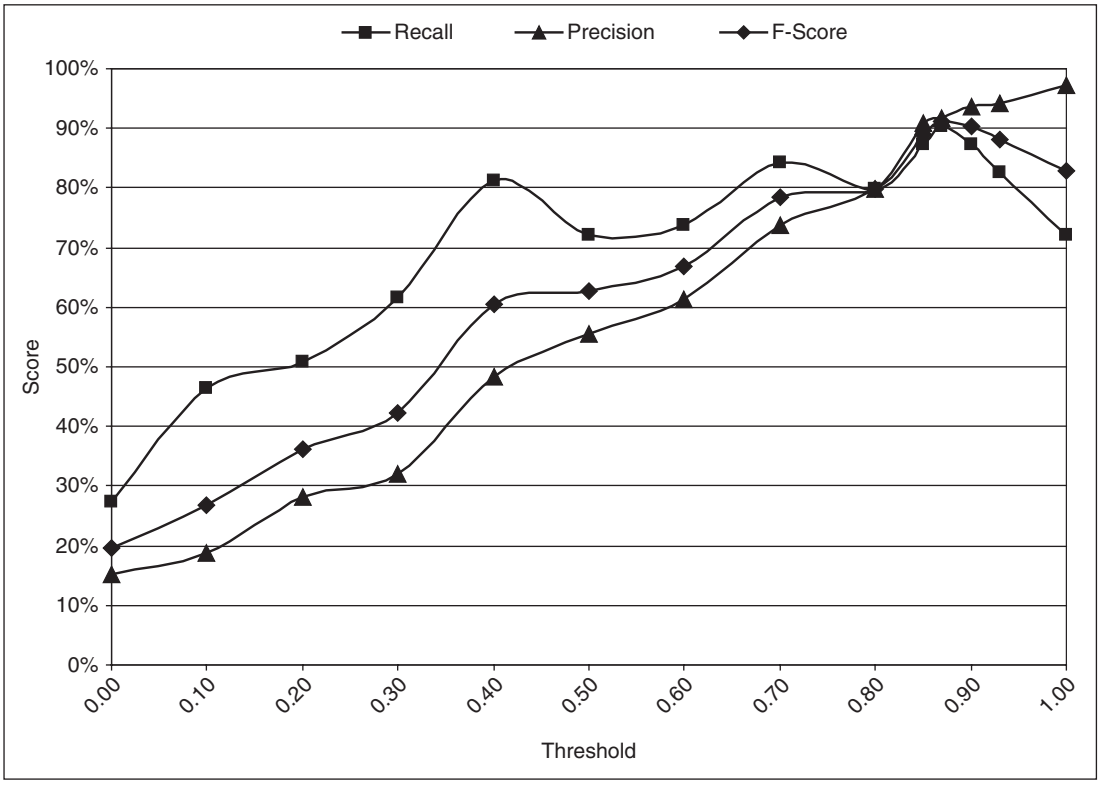

Figure 5. The observed performance of the system as the threshold parameter changes.

Table 3. Individual impact of each feature set to the system performance (I)

Deactivated feature set	Used feature sets	System performance (*F*-score (%))	Loss (%)
Lexical	Morphological, Contextual, Orthographic	81.29	9.79
Morphological	Lexical, Contextual, Orthographic	**78.82**	**12.26**
Contextual	Lexical, Morphological, Orthographic	87.90	3.18
Orthographic	Lexical, Morphological, Contextual	81.71	9.37

Each time a feature set was deactivated and the achieved system performance value was recorded. The last column shows the performance loss incurred when the specific set was not used. The loss incurred in this series is directly proportional to the impact of the deactivated feature set.

Table 4. Individual impact of each feature set to the system performance (II)

Used feature set	Deactivated feature sets	System performance (*F*-score (%))	Loss (%)
Lexical	Morphological, Contextual, Orthographic	67.21	23.87
Morphological	Lexical, Contextual, Orthographic	**69.95**	**21.13**
Contextual	Lexical, Morphological, Orthographic	56.51	34.57
Orthographic	Lexical, Morphological, Contextual	65.15	25.93

Only one feature set was used at a time. The last column shows the performance loss incurred when only that specific set and the actual token information were used. The loss incurred in this series is inversely proportional to the impact of the used feature set.

4.2.4. Generalization features. We investigated the impacts of each feature set used for the generalization process. In order to calculate the individual contribution of each set, we conducted two series of experiments. In the first series, we deactivated a feature set in each experiment, and recorded the achieved system performance. In the second series, we approached the question from a different point of view. This time, we tested the system performance by using only one feature set at a time. Tables 3 and 4 show the results of two conducted experiment series. Table 3 shows the recorded performance score and the incurred performance loss in the absence of each specific generalization feature. The biggest loss occurs (12.26%) when morphological features were not used. The individual impact of each feature set to the system performance is shown in a different way in Table 4; the recorded performance scores and the incurred performance losses were given when only one feature set and the actual token information were used at a time. The system achieved $F = 69.95\%$ using only morphological features and the actual token information. We believe that morphological features ended up being the most influential due to the agglutinative nature of Turkish.

5. Conclusions and future work

This paper presents an automatic rule learning method for identifying NEs located in Turkish texts. The presented method aims to automatically learn rules to recognize the NE patterns and generalize these patterns by processing similarities and differences between them. The system uses several generalization features to obtain an accurate generalization and to remedy the issues related to the data sparseness problem. In addition to the generalization features, an expressive rule representation language and a novel coverage algorithm are used by the system for automatic rule learning and generalization.

We performed several experiments to evaluate our method's performance. The results indicate that our suggested method can be used for extracting NEs from Turkish texts effectively. The experiments were conducted on the TurkIE corpus, generated in support of this study. The developed corpus and the corpus annotation tool are two other major contributions of this study, which will encourage and support future researchers in this area. Although there are still a few studies available, we compared the performance of the developed system with previous studies. Our system produced better results than many of the previous studies. It achieved a comparable performance score with a statistical learning system using similar resources. The system also addresses the domain adaptation problem, another key challenge in the IE field, by employing an adaptive rule learning method. The developed system minimizes the tasks requiring human intervention; it does not rely on manually developed rules/patterns. The lexical sources used in the system are kept generic to capture the variations in the patterns. Moreover, the system eliminates the burden of adding new sources by its configurable and extensible design. The impact of each generalization feature utilized was investigated from different angles. The results show that exploiting morphological features significantly improves the NER from Turkish texts due to the agglutinative nature of the language. We believe that the use of morphological features can also improve the NER performance in other agglutinative languages.

The Turkish NE recognizer is the first study of our umbrella project TurkIE, a complete IE system for Turkish. The next steps will include the development of an entity relation detector and an event extractor. Both the planned entity relation detector and the event extractor will be based on the same rule learning method. In future work, we are planning to improve the developed rule learning method by introducing correction rules before applying it to the new IE tasks. Although the use of rule exception sets explained in Section 3.5 helps to reduce false positives, handling the information regarding to the rule exceptions in a more formal way and generalizing them into correction rules would further increase the performance of the system. Moreover, we would like to see the system's behaviour in the presence of noise. Other future work will further expand the TurkIE corpus in order to examine the behaviour of our system on larger datasets. The generation of noisy data for test purposes will also be taken into consideration during the corpus expansion study.

Notes

1. Independent from the specificity of the rules, the confidence factor of all of the rules must exceed the threshold value.
2. www.cs.bilkent.edu.tr/~ilyas/TurkIE_Corpus.rar
3. www.cs.bilkent.edu.tr/~ilyas/TurkIE_Tagger.rar

References

[1] Y. Wu, J. Zhao, B. Xu and H. Yu , Chinese named entity recognition based on multiple features, *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, Morristown, NJ, 2005) 427–434.

[2] Y. Benajiba, M. Diab and P. Rosso, Arabic named entity recognition using optimized feature sets, *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 2008) 284–293.

[3] M. Rössler, Corpus-based Learning of Lexical Resources for German Named Entity Recognition, *Proceedings of LREC* (2004).

[4] A. Przepiórkowski, Slavonic information extraction and partial parsing, *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: information Extraction and Enabling Technologies* (Association for Computational Linguistics, Morristown, NJ, 2007) 1–10.

[5] N. Kushmerick and B. Thomas, Adaptive information extraction: core technologies for information agents, *Intelligent Information Agents: The AgentLink Perspective* (Springer, Berlin, 2002) 79–103.

[6] C. Siefkes and P. Siniakov, An overview and classification of adaptive approaches to information extraction, *Journal of Data Semantics* 4 (2005) 172–212.

[7] J. Turmo, A. Ageno and N. Català, Adaptive information extraction, *ACM Computing Surveys* 38(2) (2006) 4.

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

[9] D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Meyers and M. Tyson, SRI International FASTUS system: MUC-6 test results and analysis, *Proceedings of the Sixth Message Understanding Conference* (1996).

[10] R. Grishman, The NYU system for MUC-6 or where's the syntax? *Proceedings of the 6th Conference on Message Understanding Conference* (Association for Computational Linguistics, Morristown, NJ, 1995) 167–175.

[11] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell and Y.W.H. Cunningham, University of Sheffield: description of the LaSIE-II system as used for MUC-7, *Proceedings of the Seventh Message Understanding Conference* (1998).

[12] E. Riloff, An empirical study of automated dictionary construction for information extraction in three domains, *Artificial Intelligence* 85 (1996) 101–134.

[13] E. Riloff, Automatically constructing a dictionary for information extraction tasks, *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)* (1993).

[14] S. Soderland, D. Fisher, J. Aseltine and W. Lehnert, CRYSTAL: inducing a conceptual dictionary, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)* (1995).

[15] M.E. Califf and R.J. Mooney, Relational learning of pattern-match rules for information extraction, *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing* (Menlo Park, CA, 1998).

[16] M.E. Califf and R. J. Mooney, Bottom-up relational learning of pattern matching rules for information extraction, *Journal of Machine Learning Research* 4 (2003) 177–210.

[17] D. Freitag, Machine Learning for information extraction in informal domains, *Machine Learning* 39(2–3) (2000) 169–202.

[18] D. Freitag and N. Kushmerick, Boosted Wrapper Induction, *Proceedings of the Seventh National Conference on Artificial* (Austin, Texas, 2000) 577–583.

[19] C. Hsu and M. Dung, Generating finite-state transducers for semistructured data extraction from the web, *Journal of Information Systems* 23(8) (1998) 521–538.

[20] N. Kushmerick, Wrapper induction: Efficiency and expressiveness, *Artificial Intelligence* 118(1–2) (2000) 15–68.

[21] D.M. Bikel, R. Schwartz and R.M. Weischedel, An algorithm that learns what's in a name, *Machine Learning* (34) (1999) 211–232.

[22] D. Freitag and A. McCallum, Information extraction with HMMs and shrinkage, *Papers from the Sixteenth National Conference on Artificial Intelligence (AAAI-99) Workshop on Machine Learning for Information Extraction* (1999) 31–36.
[23] D. Freitag and A. McCallum, Information extraction with HMM structures learned by stochastic optimization, *Proceedings of the Seventh National Conference on Artificial* (Austin, Texas, 2000).
[24] K. Seymore, A. McCallum and R. Rosenfeld, Learning hidden Markov model structure for information extraction, *Papers from the Sixteenth National Conference on Artificial Intelligence (AAAI-99) Workshop on Machine Learning for Information Extraction* (Orlando, FL, 1999) 37–42.
[25] A. McCallum, D. Freitag and F. Pereira, Maximum entropy Markov models for information extraction and segmentation, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)* (Stanford, CA, 2000).
[26] A. McCallum and W. Li, Early results for named entity recognition with conditional random fields, fetures induction and web-enhanced lexicons, *CoNLL* (2003).
[27] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields, *Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics* (2004).
[28] H.L. Chieu and H.T. Ng, A maximum entropy approach to information extraction from semi-structured and free text, *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)* (2002) 786–791.
[29] A. Finn and N. Kushmerick, Multi-level boundary classification for information extraction, *ECML* (2004) 111–122.
[30] S. Zhao and R. Grishman, Extracting relations with integrated information using kernel methods, *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)* (2005) 419–426.
[31] E. Riloff and R. Jones, Learning dictionaries for information extraction by multi-level bootstrapping, *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence* (American Association for Artificial Intelligence, Menlo Park, CA, 1999) 474–479.
[32] R. Yangarber, R. Grishman, P. Tapanainen and S. Huttunen, Automatic acquisition of domain knowledge for Information Extraction, *Proceedings of the 18th Conference on Computational Linguistics – Volume 2, International Conference on Computational Linguistics. Association for Computational Linguistics* (Morristown, NJ, 2000) 940–946.
[33] A. Cucchiarelli and P. Velardi, Unsupervised named entity recognition using syntactic and semantic contextual evidence, *Computational Linguistics* 27(1) (2001) 123–131.
[34] I. Muslea, S. Minton and C.A. Knoblock, Selective sampling with redundant views, *AAAI/IAAI* (2000) 621–626.
[35] C.A. Thompson, M.E.Califf and R.J. Mooney, Active learning for natural language parsing and information extraction, *ICML* (1999) 406–414.
[36] R. Ghani, R. Jones, T. Mitchell and E. Riloff, Active learning for information extraction with multiple view feature sets, *20th International Conference on Machine Learning (ICML 2003)* (2003).
[37] E. Alfonseca and S. Manandhar, An unsupervised method for general named entity recognition and automated concept discovery, *Proceedings of the International Conference on General WordNet* (2002).
[38] Y. Shinyama and S. Sekine, Named entity discovery using comparable news articles, *Proceedings of the International Conference on Computational Linguistics* (2004).
[39] E.F. Tjong and Kim Sang, Introduction to the CoNLL-2002 shared task: language-independent named entity recognition, *Proceedings of the Conference on Natural Language Learning* (2002).
[40] E.F. Tjong Kim Sang and F. De Meulder, Introduction to the CoNLL-2003 shared task: language-independent named entity recognition, *Proceedings of the Conference on Natural Language Learning* (2003).
[41] H.P. Zhang, Q. Liu, H.K. Yu, Y.Q. Cheng and S. Bai, Chinese named entity recognition using role model, *Computational Linguistics and Chinese Language Processing* 8(2) (2003) 29–60.
[42] G. Fu and K. Luke, Chinese named entity recognition using lexicalized HMMs, *SIGKDD Explorer Newsletters* 7(1) (2005) 19–25.
[43] S. Sekine, R. Grishman and H. Shinnou, A decision tree method for finding and classifying names in Japanese texts, *Proceedings of the Sixth Workshop on Very Large Corpora* (1998).
[44] H. Isozaki, Japanese named entity recognition based on a simple rule generator and decision tree learning, *Proceedings of the 39th Annual Meeting on Association For Computational Linguistics* (2001) 314–321.
[45] E. Chung, Y. Hwang and M. Jang, Korean named entity recognition using HMM and CoTraining model, *Proceedings of the Sixth international Workshop on information Retrieval with Asian Languages – Volume 11* (2003) 161–167.
[46] G. Petasis, F. Vichot, F. Wolinski, G. Paliouras, V. Karkaletsis and C.D. Spyropoulos, Using machine learning to maintain rule-based named-entity recognition and classification systems, *Proceedings of the Conference of Association for Computational Linguistics* (2001).
[47] S. Boutsis, I. Demiros, V. Giouli, M. Liakata, H. Papageorgiou and S. Piperidis, A system for recognition of named entities in Greek, *Proceedings of the International Conference on Natural Language Processing* (2000).
[48] E. Bick, A named entity recognizer for Danish, *Proceedings of the Conference on Language Resources and Evaluation* (2004).
[49] P.T. Thao, T.Q. Tri, D. Dien and N. Collier, Named entity recognition in Vietnamese using classifier voting, *ACM Transactions on Asian Language Information Processing (TALIP)* 6(4) (2007) 1–18.
[50] K.S. Hasan, A. ur Rahman and V. Ng, Learning-based named entity recognition for morphologically-rich, resource-scarce languages, *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics* (2009) 354–362.

[51] J.F. Da Silva, Z. Kozareva and G.P. Lopes, Cluster analysis and classification of named entities, *Proceedings of the Conference on Language Resources and Evaluation* (2004).
[52] B. Popov, A. Kirilov, D. Maynard and D. Manov, Creation of reusable components and language resources for Named Entity Recognition in Russian, *Proceedings of the Conference on Language Resources and Evaluation* (2004).
[53] S. Katrenko and P. Adriaans, Named entity recognition for Ukrainian: a resource-light approach, *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies* (2007) 88–93.
[54] C. Grover, S. McDonald, D.N. Gearailt, V. Karkaletsis, D. Farmakiotou, G. Samaritakis, G. Petasis, M.T. Pazienza, M. Vindigni, F. Vichot and F. Wolinski, Multilingual XML-based named entity recognition for e-retail domains, *Proceedings of the 3rd International Conference on Language Resources and Evaluation* (2002).
[55] K. Saito and M. Nagata, Multi-language named-entity recognition system based on HMM, *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-Language Named Entity Recognition - Volume 15* (2003) 41–48.
[56] S. Cucerzan and D. Yarowsky, Language independent named entity recognition combining morphological and contextual evidence, *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora* (1999).
[57] G. Tur, D.H. Tur and K. Oflazer, A statistical information extraction system for Turkish, *Journal of Natural Language Engineering* 9(2) (2003) 181–210.
[58] O. Bayraktar, T.T. Temizel, Person name extraction from Turkish financial news text using local grammar based approach, *Proceedings of the International Symposium on Computer and Information Sciences* (2008).
[59] D. Kucuk and A. Yazici, Named entity recognition experiments on Turkish texts, *Proceedings of the 8th international Conference on Flexible Query Answering Systems* (2009) 524–535.
[60] K. Oflazer, Two-level description of Turkish morphology, *Literary and Linguistic Computing* 9(2) (1994) 175–198.
[61] J. Hankamer, Morphological parsing and the lexicon, *Lexical Representation and Process* (1989) 392–408.
[62] I. Cicekli and N.K. Cicekli, Generalizing predicates with string arguments, *Applied Intelligence* 25(1) (2006) 23–36.
[63] T. Daybelge and I. Cicekli, A rule-based morphological disambiguator for Turkish, *Proceedings of Recent Advances in Natural Language Processing* (2007) 145–149.
[64] M. Kutlu, Noun phrase chunker for Turkish using dependency parser, M.S. Thesis, Bilkent University, July 2010.
[65] N. Chinchor. MUC-7 named entity task definition, version 3.5, *Proceedings of the Seventh Message Understanding Conference* (1998)