Persian Named Entity Recognition based with Local Filters

Morteza Kolali Khormuji
Young Researchers and Elite Club, Islamic Azad University, Bushehr Branch, Bushehr, Iran

Mehrnoosh Bazrafkan
Department of Computer, Islamic Azad University, Science and Research Branch-Bushehr, Bushehr, Iran

ABSTRACT

$Persian\ (Farsi)\ language$ named entity recognition is a challenging, difficult, yet important task in natural language processing. This paper presents an approach based on a $Local\ Filters$ model to recognize $Persian\ (Farsi)\ language$ named entities. It uses multiple dictionaries, which are freely available on the Web. A dictionary is a collection of phrases that describe named entities. The framework is composed of two stages: (1) detection of named entity candidates using dictionaries for lookups and (2) filtering of false positives based. Dictionary lookups are performed using an efficient prefix-tree data structure. Our $dictionary - based$ recognizer performs on $Persian\ (Farsi)\ language$ with up to 88.95% precision, 79.65% recall, and an 82.73% F1 score using ASEM.

General Terms:

Natural language Processing, Named Entity Recognition

Keywords:

Persian Natural language processing, Named Entity Recognition, Local Filters

1. INTRODUCTION

Named entity recognition (NER) is widely acknowledged as one of the central tasks in natural language processing (NLP). The essential goal of NER is to identify and classify certain proper nouns, such as person names (PER), organizations (ORG), locations (LOC), and so on. NER has attracted much attention in the research community for a long time. A named entity (NE) is a phrase representing an item of a class.

So far many corpora have been developed in other languages and base on them several different models and methods have been applied for Named entity recognition (NER). The models and methods can be divided into two main approaches: the first one obeys a statistical approach which utilizes annotated corpora and the second one is the rule-based non-statistical approach which is based on machine learning and human knowledge. Some reported methods are as follow: Semi-supervised Named Entity Recognition [10, 2, 3, 4], Building a Corpus-Derived Gazetteer [11], An Approach for Named Entity Recognition in Poorly Structured Data[12], Chinese and Disambiguation Based on Wikipedia[13], Resolution in Legal Text[14], Incorporating Linguistic Expertise Using ILP[15], Multiobjective Optimization Approach [16], Hidden Markov Models [17, 1].

This work presents a dictionary-based NER framework. Our dictionary-based recognizer is composed of two stages. In the first stage, an input text and available dictionaries are prepared for a dictionary-based NE detector. Then, the detector finds candidates for NES and their positions in the input text by matching the string in the dictionary against the text. In the filtering stage, false positives are removed. The local filters, together with the part-of-speech filter, are used to improve the quality of the NE recognition by filtering out noisy matches of the first stage.

In the task of named entity classification (NEC), a class or type is assigned to an NE. For instance, the "IRAN" classes for an NEC task are person, location, organization, and miscellaneous.

2. NAMED ENTITY RECOGNITION

Named Entity Recognition Systems have been created that use linguistic grammar-based techniques and statistical models. Handcrafted grammar-based systems are usually obtained better precision, however, lower recall in months of work by experienced linguists cost calculation. Statistical NER systems typically require a large amount of manually annotated training data. usually find the sequence of tags that maximizes the probability $p(N|S)$,where S is the sequence of words in a sentence, and N is the sequence of named-entity tags assigned to the words in [5].

NER are sequence of words, which refer to $persons$, $organizations$, $locations$, etc. These types can be more fine-grained. For example, the class $person$ can be subdivided into several subclasses like $(pesaran)$ which means $boys$. The actual type hierarchy can be simple or more complex like trees or even graphs. NER are defined to include numerical expressions, such as money or dates; however, we do not include them.

2.1 Features

Our integration is done by feeding the output of the rule-based system as features to machine-learning classifiers. We call these features the rule-based features. Most of NER systems relay a set of feature functions that represent a machine-readable characteristic

of a word. Choice of features for a NER system is the most important aspect of any NER system. Now we consider most popular features for a NER system and their types. These three categories of features are the ones mainly used for existing systems:

(word features) $case$: (AllCaps, Capitalized, MixCase). $morphology$: (prefix, suffix, stem). PoS: (proper name, common noun, foreign word). $function$: (n-gram, length, lower-case).

(dictionaries:) $general\ \ gazetteers$: (stop words, capitalized nouns, abbreviations). $entities$: (locations, organizations, first names, last names). $entity\ cues$:

(document features include $metadata$ of the document:) $local\ \ syntax$: (position in sentence, in document). $meta\ \ information$: (URI, lists, tables).

2.2 Dictionaries

Dictionary or gazetteer is a collection of words or phrases referring to a particular entity. For example, the phrase "$Bushehr$" appearing in the dictionary with location names refers to the city in Iran, whereas the word "$Aghayi$" which means $Mr.$ refers to the list of role names for persons. A dictionary can contain instances of one type like locations, or several types. Dictionaries of names will be used as features in trained approaches or a way to identify all candidates in a given text. Dictionaries provide powerful features that improve the performance of the NER and NEC systems. Commonly used dictionaries for NER can be created from:

(Wikipedia) (e.g. titles of the pages, anchor texts).

(knowledge bases like) (Persian corpus named BIJANKHAN [6], DBpedia [7])

Lookup Techniques: The lookup technique is a crucial part of a system that uses dictionaries. Main techniques for making lookups in dictionaries are: $fullmatching$, matching based on stemming or $lemmatization$, matching that uses $Soundex$, $approximate\ matching$.

Exact Matching: It is a type of matching whereby a dictionary entry is exactly matched against a text. However, phrases in a text having different word forms cannot be matched.

Lemmatization: In contrast to stemming, lemmatization strives to improve the problems where stemmer fails. Lemmatization is a process of determining the basis form of a word.

Soundex: As an alternative to stemming and lemmatization, the Soundex algorithm provides a string modi
cation technique. Soundex is a word modi
cation technique that converts a word into its sound as uttered in English. Namely, words are decoded such that the similarly spelled or pronounced words build the same code. For example, the $Soundex$ of the words "$Iranian$" and "$Iran$" is "$IR48$".

Approximate matching: The words in a text can be written with typos. An approximate matching is a method of finding similar strings for a given pattern. An example of such operations are: $insertion$ of a single letter into a string, $deletion$ of a single letter from a string, $replacement$ of a letter from a string by the new letter. Typically, approximate string matching algorithms use the edit distance that equals or is less than two. However, computing approximate matches for huge dictionaries is a time consuming step.

Fig. 1. An example of dictionaries with phrases.

3. DICTIONARY-BASED NER

In this section, we describe our approach to dictionary-based named entity recognition. A named entity is a phrase representing an item of a class. A dictionary is a collection of phrases that describe named entities. For example, the phrase "$Bashgah\ \ Pajoheshgaran\ \ Javan$" or "$BPJ$" is a named entity of the class "$Organization$".

The dictionary-based recognizer is composed of two stages. In the first stage, an input text and available dictionaries are prepared for a dictionary-based named entity detector. The detector finds candidates for named entities and their positions in the input text by matching the string in the dictionary against an input text. In the second stage, NE candidates representing false positives are filtered out. Furthermore, the system performs postprocessing of the NER result like re-finding missed NER and combining adjacent NER into a single entity.

3.1 Creating Dictionaries

An essential component of the DB-NER is dictionaries. The dictionaries for the DBNER tool can be created from different sources. In our work, we use entity repositories provided by the National Library and Archives Organisation of Iran (NLAI).

3.1.1 National Library and Archives Organisation of Iran $NLAI$. Nowadays, many structured Web resources are available in RDF (Resource Description Framework) format. The National Library and Archives Organisation provides such a resource for Iran. The library provides repositories of the entities for several classes like "$person$", "$location$", "$organization$" and "$article\ \ titles$" like "$Ayandih\ \ hosh\ \ masnoei$". Therefore, it is easy to create typed dictionaries from the library repositories. As a result, not only entities, but also the titles of articles can be found by a NER system that uses such kind of source as dictionaries. Additionally, the fields with first names, last names, family names, and full names are used to create the dictionaries for person names. A small example of such dictionaries is shown in Figure 1.

3.2 Text Preprocessing

In the preprocessing step, an input text as well as dictionaries are prepared for the NE candidates detector . The preprocessing is an essential step needed by our DB-NER, because:

(1) NEs appearing in the input text can differ from the dictionary phrases.

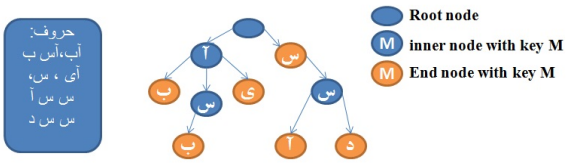

Fig. 2. An example of a prefix tree containing six words.

(2) raw dictionaries often contain noisy entries like phrases consisting only of stop words or symbols.

3.2.1 Tokenization. The tokennizer splits the text into very simple tokens such as numbers, punctuation and words of different types. In the default set of rules, the following kinds of Token and SpaceToken are possible: word; number; symbol; Punctuation; SpaceToken. Also, there is an English Tokeniser in this system. It is a processing resource that comprises a normal tokeniser and a JAPE transducer. The transducer has the role of adapting the generic output of the tokeniser to the requirements of the English part-of-speech tagger.

languages have different punctuation styles. For example, compare the date 1393/04/15 in Persian and the same date 09/06/2014 in English. As a result, the word tokenization is a language-dependent task and is usually solved using hand-crafted rules or system trained on manually tokenized texts. As a result, the word tokenization is able to disambiguate part-of-word punctuation and end-of-sentence punctuation. One of the first steps for keeping the dictionaries and the text in an unified form is tokenization.

3.3 Dictionaries Preprocessing

Each dictionary phrase is tokenized, conflated, and optionally modified as described in previous Section. However, most dictionaries suffer from noise. Therefore, cleaning of dictionaries is performed before the dictionaries are used by a NE candidates detector. In our approach, the phrases consisting only of stop words are removed. Additionally, a disambiguating term in parentheses of a phrase is removed.

3.3.1 Efficient Storage and Reading of Dictionaries. Since the dictionaries used by DB-NER are large and cannot fit into the main memory of a desktop machine, we need a technique for efficient storing and reading of the dictionaries.

The Berkeley DB [8] allows us to store and process different operations over millions of phrases in the dictionaries efficiently. In comparison to relational databases, the Berkeley DB (BDB) has a simpler interface for data management. The BDB API is implemented in the form of ordered key-value storage, which is partially loaded (cached) from the hard drive into the main memory. Therefore, the BDB is an appropriate choice for our DB-NER framework, which allows running DB-NER on a desktop machine.

3.3.2 Prefix Tree. To allow efficient lookups into dictionaries, a prefix-tree data structure is used. The prefix tree, known as trie, is a general tree where each node represents a prefix of a string by traversing from the root to the node. The root of a trie denotes an empty prefix. Figure 2 shows an example of the prefix tree.

3.3.3 Token Trie. Each phrase in a dictionary is a sequence of tokens. For space and time efficiency, each node in a prefix tree is encoded as a single token in Figure 3. Such a modification of a prefix tree is called a token trie. To manage huge dictionaries, the

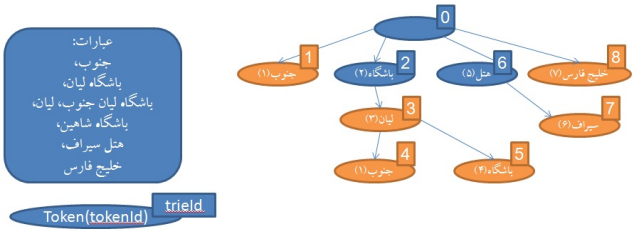

Fig. 3. An example of a token trie implemented via Berkeley DBs key-value storage.

implementation of the token trie uses ordered key-value storage. A token trie is constructed as follows:

(1) each token has an unique integer identifier tokenId.
(2) each node of a trie has an unique integer identifier trieId.
(3) root of a trie has zero as an identifier.
(4) root of the trie does not refer to any token.

3.4 Candidates Detection

Given prepared dictionaries and an input text, the DB-NER searches for NE candidates in the text. Specifically, for each dictionary, the system tries to find the longest dictionary match of the text snippet starting from the first token. To this end, the system proceeds to compute the longest match starting from the next token. As a result, a list of NE candidates for a dictionary is found. In this way, lists of NE candidates are computed for each dictionary.

3.4.1 Partial Labeling Algorithm. Our high-precision partial labeling algorithm is presented as Algorithm 1. To label a sentence s, we first recognize candidate entities in the sentence. $Candidate\ entities$ consist of sequences of capitalized words. However, we also use statistics drawn from web text to decide if candidate entities that are connected by $connectorwords$ $(va, barayi, az, dar)$ which means (of, for, the, and, in) should be joined into a single candidate by using an independence test with a manually chosen threshold. For example, "$Islamic\ Republic\ of\ Iran$" should be one candidate entity, but "$Europe\ and\ Asia$" should be two. All tokens that are not part of candidate entities are labeled with $'O'$ labels, indicating nonentity tokens. Next, we consider each candidate entity. The $FULLYCOVERED$ function tests whether or not all tokens in a candidate entity can be labeled using the gazetteers. If so, the candidate entity is labeled according to the classes in the gazetteers. If any tokens cannot be labeled with gazetteer matches, then the whole candidate is marked with $'UNK'$ labels, indicating that we do not know the label of those tokens.

3.4.2 Exact and fuzzy phrase search. The system uses either an exact or a fuzzy phrase search to check if the text contains a dictionary phrase. In the exact phrase search, an either original or modified string of text and dictionaries are used. The fuzzy phrase search strategy employs the original, unmodified text and dictionaries.

3.4.3 Exact Phrase Search. In the exact phrase search, each dictionary is kept in a cacheable out-memory key-value storage in the form of a token trie. A match is found if the tokens of a phrase in a text snippet and in the token trie match exactly. In our implementation, tokens of the input text and dictionary phrases undergo string modifications as introduced in previous Section The match is defined over the modified tokens.

Table 1. **Algorithm 1**: The dictionary-driven partial labeling algorithm.

Input: Sentence s, dictionary D
Output: Labels for s
C = $\mathcal{R}$ECOGNIZE **C**ANDIDATE **E**NTITIES(s);
LABEL NON ENTITY TOKENS($'O'$);
foreach $Candidate\ c \in C$ **do**
. **if** $FULLYCOVERED(c; D)$ **then**
. $LABELWITHGAZETTEERS(c; D)$;
. **else**
. $LABEL(c, 'UNK')$;
. **end**
end

Table 2. An example sentence of Bijankhan dataset for Persian (Farsi) language. The BIO tags have following format: "B" for first token of an NE, "I" for inter token of an NE, and "O" for not an NE. The named entity tag "O" denotes none of the four classes (person, location, organization, miscellaneous).

token	named entity	BIO tag	named entity tag
AvA	AvA kolali	B	Person
kolali	AvA kolali	I	Person
in			O
Bushehr	Bushehr	B	Location

3.4.4 Fuzzy Phrase Search. Fuzzy phrase searches use an approximate string matching algorithm that can ignore the suffixes of each word in a phrase. The char trie data structure used by fuzzy search allows partial phrase matching as well. In particular, the first letters of a word are matched exactly, whereas the remaining part of the word is ignored.

3.5 Candidates Filtering

Dictionaries used in a DB-NER may contain noisy phrases. Therefore, the result of the detection step will contain a large number of incorrect NE candidates. A PoS filtering step is performed to remove false positives.

3.5.1 Part-of-Speech Filtering. Dictionaries contain ambiguous words. One of the solutions is to filter out all ambiguous words. However, it leads to a low recall and additional effort needed for checking the words for ambiguity, which is not straightforward. Specifically, the DB-NER system allows the following PoS filters: Noun-Filter and Proper-Noun-Filter.

3.5.2 Local Filtering. The PoS filtering can fail in some cases and filter out correctly named entities from the resulting set. There are three frequent reasons for this incorrect filtering. First, the PoS tagger can make wrong suggestions. Second, NE candidates do not contain nouns. Therefore, the PoS filter removes these NE candidates.

3.6 Postprocessing

The resulting output of the filtering step is post processed as follows:

(1) reactivating of NEs that are filtered out by the PoS filter.
(2) combining of adjacent NEs.
(3) merging of filtered lists with NEs into final list of NEs.

Table 3. Dictionaries used by the $DB-NER$. The AEM value is not defined for role names, because role names do not refer to any of entities.

dictionary name	unique phrases	AEM
First names	1.232.004	6.1
Last names	1.176.221	5.2
Full name	9.288.600	1.1
Locations	256.700	1.3
Organizations	1.389.287	1.6
Miscellaneous	328.390	1.4
Products	14.317	1.2

4. EXPERIMENTAL RESULTS

In this section, we describe the configuration of the dictionary-based named entity recognizer. We introduce the data set for evaluation of DB-NER and the PoS tagger it uses.

4.1 Bijankhan Dataset

The Bijankhan corpus [9] is a tagged corpus that is suitable for natural language processing research on the Persian (Farsi) language. The four main NE types used in the dataset are persons, locations, organizations, and miscellaneous (everything else). The Bijankhan corpus is subdivided into the following sets:

(1) a training set to learn the parameters of a NER system.
(2) a development set for tuning of system parameters.
(3) a test set for comparison of the NER systems.
(4) unlabeled set for different purposes.

In our experiments, we use a slight modification of the original format used in the Bijankhan dataset. Our format has the structure as depicted in Table 2.

4.2 Dictionaries

DB-NER uses dictionaries created from the National Library and Archives Organisation of Iran (NLAI). For evaluation of the system, we use several typed dictionaries that are listed in Table 3. The dictionary with first names has the largest average number of entities per named entity or mention (AEM) among used dictionaries. For example, the name "$Morteza$" refers to 65.397 entities. In contrast to first names, names of products are less ambiguous in GNL, such that only 1.2 entities on average are connected with a single product name.

4.3 Evaluation Methodology

We consider the usual metrics for the evaluation of a NER system. Since NER is a subtask of natural language processing, the evaluation of the NER system must be performed from the point of view of human linguists. The following aspects are important for such an evaluation:

(1) boundaries of the named entities
(2) type labels of the named entities
(3) positions of the named entities

There are different guidelines for evaluation of a NER system:

MUC (the message understanding conference). For the MUC evaluation metric, the score is based on $correct\ type\ detection$ and $exact\ match\ detection$.

Table 4. Comparison of Stanford's NER and different configurations of the $DBNER$.

	Stanford's NER				
Setting	Evaluation Metric	Precision	Recall	F1	AEM
Stanford	ESEM	68.02%	58.04%	**61.37%**	∅
Stanford	ASEM	81.35%	68.01%	**72.82%**	∅
	Dictionary-Based NER: exact phrase search				
Setting	Evaluation Metric	Precision	Recall	F1	AEM
$ORIG_{NNF}$	ASEM	30.84%	79.00%	43.37%	306
$ORIG_{PNF}$	ASEM	79.33%	60.40%	72.82%	701
$ORIG_{PNF,LF}$	ASEM	31.33%	79.40%	43.82%	211
$ORIG_{PNF,LF}$	ASEM	78.33%	72.40%	81.82%	340
$LEMM_{PNF,LF}$	ASEM	88.95%	79.65%	**82.73%**	375
$LEMM_{PNF,LF}$	ESEM	84.86%	71.40%	**72.70%**	340
	Dictionary-Based NER: fuzzy phrase search				
Setting	Evaluation Metric	Precision	Recall	F1	AEM
$RATIO0.8_{PNF,LF}$	ESEM	73.02%	68.04%	**70.37%**	480
$RATIO0.8_{PNF,LF}$	ASEM	86.35%	78.01%	**80.82%**	498

CoNLL (the conference on natural language learning). Exact matching is used for the CoNLL scoring schema. Moreover, the precision is defined as the ratio between the correct returned named entities and all returned named entities.

ACE (automatic content extraction). The last evaluation strategy is ACE, which has several parameters for setting priorities of evaluation. In an ACE metric, evaluation weights are defined for all classes of named entities.

Exact and approximate string matching metrics The usual metrics are used for evaluation of NER and NEC systems. Since the NEC task is out of scope for our work, we ignore types of the named entities. Therefore, for evaluation of our DB-NER framework, we use the following metrics:

(1) exact string evaluation metric (**ESEM**)
(2) approximate string evaluation metric (**ASEM**)

The exact string evaluation metric is similar to the metric introduced by the CoNLL community. Specifically, the boundaries and positions of the named entities are considered.

By the approximate string evaluation metric, we relax one of the aspects above. In particular, the detected named entity that has (partial) overlapping with the NE from the GT is thereby treated by the ASEM as true positive.

We present an evaluation of the DB-NER. The experimental results are depicted in Table 4. The remainder of the section uses the following notations:

(1) $ORIG$ (exact matching of original texts and dictionaries), $LEMM$ (lemmatization), $LIFT$ (lifting).
(2) $RATIO\ X$ is a method of dictionary lookups that is based on fuzzy phrase search.
(3) NNF and PNF are Noun-Filter and Proper-Noun-Filter.
(4) LF is a shortcut for local filters.

The best results of DB-NER are achieved by combining the Proper-Noun-Filter with the local filtering. The setting that uses lemmatized input text and lemmatized dictionaries achieves an 88.95% precision, 79.65% recall, and an 82.73% F1 score using ASEM. The result of the DB-NER configuration with lemmatization, local filters, and the Proper-Noun-Filter for DB-NER, provides a higher precision, recall, and F1 score than the result achieved by Stanford's NER framework. However, the configuration based on lemma computation introduce additional dependency because lemmatization is language specific.

5. CONCLUSIONS

An approach based on a $Local\ Filters$ model to recognize $Persian\ (Farsi)\ language$ named entities. We have presented an approach for named entity recognition that uses dictionaries to detect named entity candidates to filter out false positives. Our dictionary-based recognizer is capable to work efficiently with millions of phrases in each dictionary due to a cacheable prefix-tree data structure, which is used for dictionary lookups. Our recognizer uses multiple dictionaries created from the entities of National Library and Archives Organisation of Iran (NLAI). Therefore, we tune our system to Persian Iran. In addition, Persian is a more expressive language and more difficult than English, because it has a complex morphology. In contrast to the prior works that are based on machine learning methods, our dictionary-based recognizer does not need labeled training data.

Our $DB-NER$ performs on $Persian\ (Farsi)\ language$ with up to 88.95% precision, 79.65% recall, and an 82.73% F1 score using ASEM. This result was achieved by the setting, which uses lemmatization as a dictionary lookup technique and applies with local filters.

6. REFERENCES

[1] Branimir T. Todorovi, Svetozar R. Rancic, Edin H. Mulalic,Context Hidden Markov Model for Named Entity Recognition, Approximation and Computation, springer, Volume 42, 2011, pp 447-460.

[2] Duc-Thuan Vo, Cheol-Young Ock, A Hybrid Approach of Pattern Extraction and Semi-supervised Learning for Vietnamese

Named Entity Recognition, Computational Collective Intelligence. Technologies and Applications, springer, Volume 7653, 2012, pp 83-93.

[3] David Nadeau, Peter D. Turney, Stan Matwin, Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision, 2007 http://citeseerx.ist.psu.edu/, doi=10.1.1.109.4327.

[4] Thi-Ngan Pham Vietnamese People's Police Acad., Hanoi, Vietnam, Le Minh Nguyen, Quang-Thuy Ha, Named Entity Recognition for Vietnamese Documents Using Semi-supervised Learning Method of CRFs with Generalized Expectation Criteria, Asian Language Processing (IALP), 2012 International Conference on, IEEE, 13-15 Nov. 2012,Page 85 - 88.

[5] Chieu, Hai Leong and Ng, Hwee Tou, Named Entity Recognition: A Maximum Entropy Approach Using Global Information, COLING'02: Proceedings of the 19th international conference on Computational linguistics, 2002.

[6] M. BIJANKHAN, The Role of the Corpus in Writing a Grammar: An Introduction to a Software, Iranian Journal of Linguistics, vol. 19, no. 2, fall and winter 2004.

[7] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Soren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. Web Semantics, 7(3):154-165, 2009.

[8] Michael A. Olson, Keith Bostic, and Margo Seltzer. Berkeley DB. In ATEC '99: Proceedings of the annual conference on USENIX Annual Technical Conference, pages 43-43, Monterey, California, 1999.

[9] http://ece.ut.ac.ir/dbrg/bijankhan/ Bijankhan corpus was created in DBRG Lab. at University of Tehran ECE department.

[10] Rathany Chan Sam, Huong Thanh Le, Thuy Thanh Nguyen, Thien Huu Nguyen, "Combining Proper Name-Coreference with Conditional Random Fields for Semi-supervised Named Entity Recognition in Vietnamese Text", Advances in Knowledge Discovery and Data Mining, springer Volume 6634, 2011, pp 512-524.

[11] Norshuhani Zamin, Alan Oxley, Building a Corpus-Derived Gazetteer for Named Entity Recognition, Software Engineering and Computer Systems, springer, Volume 180, 2011, pp 73-80.

[12] Nuno Freire, Jos Borbinha, Pvel Calado, An Approach for Named Entity Recognition in Poorly Structured Data, The Semantic Web: Research and Applications, springer, Volume 7295, 2012, pp 718-732.

[13] Yu Miao, Lv Yajuan, Liu Qun, Su Jinsong, Xiong Hao, Chinese Named Entity Recognition and Disambiguation Based on Wikipedia, Natural Language Processing and Chinese Computing, springer, Volume 333, 2012, pp 272-283.

[14] Christopher Dozier, Ravikumar Kondadadi, Marc Light, Arun Vachher, Sriharsha Veeramachaneni, Ramdev Wudali, Named Entity Recognition and Resolution in Legal Text, Semantic Processing of Legal Texts, springer, Volume 6036, 2010, pp 27-43.

[15] Anup Patel, Ganesh Ramakrishnan, Pushpak Bhattacharya, Incorporating Linguistic Expertise Using ILP for Named Entity Recognition in Data Hungry Indian Languages, Inductive Logic Programming, springer, Volume 5989, 2010, pp 178-185.

[16] Asif Ekbal, Sriparna Saha, Christoph S. Garbe, Multiobjective Optimization Approach for Named Entity Recognition, PRICAI 2010: Trends in Artificial Intelligence, springer, Volume 6230, 2010, pp 52-63.

[17] Micha Marcinczuk, Maciej Piasecki, Study on Named Entity Recognition for Polish Based on Hidden Markov Models, Text, Speech and Dialogue, springer, Volume 6231, 2010, pp 142-149.