Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity

David Nadeau[1,2], Peter D. Turney[1] and Stan Matwin[2,3]

[1] Institute for Information Technology
National Research Council Canada
{david.nadeau, peter.turney}@nrc-cnrc.gc.ca
[2] School of Information Technology and Engineering, University of Ottawa
{dnadeau, stan}@site.uottawa.ca
[3]Institute for Computer Science, Polish Academy of Sciences

Abstract. In this paper, we propose a named-entity recognition (NER) system that addresses two major limitations frequently discussed in the field. First, the system requires no human intervention such as manually labeling training data or creating gazetteers. Second, the system can handle more than the three classical named-entity types (person, location, and organization). We describe the system's architecture and compare its performance with a supervised system. We experimentally evaluate the system on a standard corpus, with the three classical named-entity types, and also on a new corpus, with a new named-entity type (car brands).

1 Introduction

This paper builds on past work in unsupervised named-entity recognition (NER) by Collins and Singer [3] and Etzioni *et al.* [4]. Our goal is to create a system that can recognize named-entities in a given document without prior training (supervised learning) or manually constructed gazetteers. (We use the term *gazetteer* interchangeably with the term *named-entity list*.)

Collins and Singer's [3] system exploits a large corpus to create a generic list of proper names (named-entities of arbitrary and unknown types). Proper names are collected by looking for syntactic patterns with precise properties. For instance, a proper name is a sequence of consecutive words, within a noun phrase, that are tagged as NNP or NNPS by a part-of-speech tagger and in which the last word is identified as the head of the noun phrase. Like Collins and Singer, we use a large corpus to create lists of named-entities, but we present a technique that can exploit diverse types of text, including text without proper grammatical sentences, such as tables and lists (marked up with HTML).

Etzioni *et al.* [4] refer to their algorithm as a named-entity *extraction* system. It is not intended for named-entity *recognition*. In other words, it is used to create large lists of named-entities, but it is not designed for resolving ambiguity in a given document. The distinction between these tasks is important. It might seem that having a list of entities in hand makes NER trivial. One can extract city names from a given document by merely searching in the document for each city name in a city list. However, this strategy often fails because of ambiguity. For example, consider the words “It” (a city in Mississippi State and a pronoun) and “Jobs” (a person’s surname and a common noun). The task addressed by Etzioni *et al.* could be called *automatic gazetteer generation*. Without ambiguity resolution, their system cannot perform robust, accurate NER. This claim is supported by the experiments we present in Section 3.

In this paper, we propose a named-entity recognition system that combines named-entity extraction (inspired by Etzioni *et al.*[4]) with a simple form of named-entity disambiguation. We use some simple yet highly effective heuristics, based on the work of Mikheev [9], Petasis *et al.* [13], and Palmer and Day [12], to perform named-entity disambiguation. We compare the performance of our unsupervised system with that of a basic supervised system, using the MUC 7 NER corpus [1]. We also show that our technique is general enough to be applied to other named-entity types, such as car brands, or bridge names. To support this claim, we include an experiment with car brands.

The paper is divided as follows. First, we present the system architecture in Section 2. Then, we compare its performance with a supervised baseline system on the MUC 7 NER corpus in Section 3. Next, we show that the system can handle other type of entities, in addition to the classic three (person, location, and organization), in Section 4. We discuss the degree of supervision in Section 5. We conclude in Section 6 by arguing that our system advances the state-of-the-art of NER by avoiding the need for supervision and by handling novel types of named-entities. The system’s source code is available under the GPL license at http://balie.sourceforge.net.

2 Unsupervised Named-Entity Recognition System

The system is made of two modules. The first one is used to create large gazetteers of entities, such as a list of cities. The second module uses simple heuristics to identify and classify entities in the context of a given document (i.e., entity disambiguation).

2.1 Generating Gazetteers

The task of automatically generating lists of entities has been investigated by several researchers. In Hearst [6], lexical patterns are studied that can be used to identify nouns from the same semantic class. For instance, a noun phrase that follows the pattern “the city of” is usually a city. In Riloff and Jones [14], a small set of lexical patterns and a small set of entities are grown using mutual bootstrapping. Finally, Lin and Pantel [7] show how to create large clusters of semantically related words using an unsupervised technique. Their idea is based on examining words with similar syntactic dependency relationships. They show they can induce semantic classes such as car brands, drugs, and provinces. However, their technique does not discover the labels of the semantic classes, which is a common limitation of clustering techniques.

The algorithm of Etzioni *et al.* [4] outperforms all previous methods for the task of creating a large list for a given type of entity or semantic class; the task of automatic gazetteer generation. Nadeau [11] shows that it is possible to create accurate lists of cities and car brands in an unsupervised manner, limiting the supervision to a seed of four examples. In the remainder of this section, we summarize how to generate a list of thousands of cities from a seed of a few examples, in two steps (repeated if necessary).

2.1.1 Retrieve Pages with Seed

The first step is information retrieval from the Web. A query is created by conjoining a seed of *k* manually generated entities (e.g., “Montreal” AND “Boston” AND “Paris” AND “Mexico City”). In our experience, when *k* is set to 4 (as suggested by Etzioni *et al.* [4]) and the seed entities are common city names, the query typically retrieves Web pages that contain many names of cities, in addition to the seed names. The basic idea of the algorithm is to extract these additional city names from each retrieved Web page.

The same strategy can be applied to person names, company names, car brands, and many other types of entities. Although it is outside of the scope of this paper, we should mention that we successfully applied this technique to more than 50 named-entity types.

2.1.2 Apply Web Page Wrapper

A Web page wrapper is a rule-based system that identifies the location of specific types of information within a Web page. For example, a wrapper for identifying the location of news headers on the Web site *radio-canada.ca* might contain the rule, “A header is an HTML node of type <a>, with text length between 10 and 30 characters, in a table of depth 5 and with at least 3 other nodes in the page that satisfy the same rule.”

The gazetteer generation algorithm proceeds by learning rules that identify the locations of positive examples. For each page found in 2.1.1, a Web page wrapper is trained on the *k* positive examples that are known to appear in the page, but only if they are strictly contained in an HTML node (e.g., <td> Boston </td>) or surrounded by a small amount of text inside an HTML node (e.g., <td> Boston hotel </td>). The remaining HTML nodes in the page are treated as if they were negative examples, but we only include in the negative set the nodes with the same HTML tags as the positive examples [11]. For instance, if the *k* positive nodes are tagged as bold (i.e., “<b>”), then the negative examples will be restricted to the remaining bold text in the Web page. The Web page wrapper we used is similar to Cohen and Fan’s [2] wrapper, in terms of the learning algorithm and the feature vector.

As described above, Web page wrapping is a classification problem. A supervised learning algorithm is used to classify unknown entities in the current Web page. In this application, the training set and the testing set are the same. The learning algorithm is trained on the given Web page and then the learned model is applied to reclassify the text in the same Web page. The idea is to learn rules, during training, that identify the locations of the known entities (the seed entities) and can be applied, during testing, to identify entities appearing in similar contexts, which may be further positive examples.

Two main problems make this task difficult. First, there is noise in the class labels in the training data, because everything except the seed words are initially labeled as negative. If the page contains more than *k* entities of the desired type, the very nodes we want to extract were labeled as negative. The second problem is the class imbalance in the data. Along with the *k* positive examples, there are usually hundreds or thousands of negative examples. These two problems are handled by noise filtering and cost-sensitive classification, respectively.

At this point, our technique goes beyond the system of Etzioni *et al.* [4], which uses a simple Web page wrapper, consisting of hand-crafted rules. To handle the problem of noise in the class labels, we use a filtering approach inspired by Zhu *et al.* [16]. The noise filtering strategy is to simply remove any instance similar to a positive instance. We say that two nodes are similar when their feature vectors are identical, except for the text length feature. (Refer to Cohen and Fan [2] for a description of the Web page wrapper's features.) Using this filter, an average of 42% of the examples that are initially labeled as negative are removed from the training set. These examples are left in the (unlabeled) testing set. When the trained model is later applied to the testing set, some of the removed examples may be classified as positive and some may be classified as negative.

To handle the class imbalance problem, we use a cost-sensitive supervised learning system. Using the original unbalanced dataset, the wrapper is almost incapable of extracting new entities. It mainly guesses the majority class (negative) and only extracts the initial seed from Web pages. To discourage the learning algorithm from using the trivial solution of always guessing the majority class, a high cost is assigned to misclassification errors in which a positive example is classified as negative. This cost-sensitive approach over-samples the positive examples to rebalance the dataset. This rebalancing must be done for each individual Web page, to take into account the imbalance ratio of each wrapper. Rebalancing is performed automatically, by randomly choosing HTML nodes to add to the dataset, up to the desired ratio of positive to negative examples.

Past research suggests that supervised learning algorithms work best when the ratio is near 1:1 [8]. We hypothesized that the wrapper would work best when we rebalanced the dataset by duplicating positive instances until the ratio reached 1:1. To verify this hypothesis, we studied the behavior of the wrapper with different ratios on a set of 40 Web pages. As expected, we found that the wrapper performance is optimal when the ratio is 1:1. We therefore use this ratio in the experiments in Sections 3 and 4.

2.1.3 Repeat

The two steps above (2.2.1, 2.2.2) are repeated as needed. Each iteration brings new entities that are added to the final gazetteer. At each iteration, *k* new randomly chosen entities are used to refresh the seed for the system. Entities are chosen from the gazetteer under construction. Preference is given to seed entities that are less likely to be noise, such as those appearing in multiple Web pages.

2.2 Resolving Ambiguity

The *list lookup strategy* is the method of performing NER by scanning through a given input document, looking for terms that match a list entry. The list lookup strategy suffers from three main problems: (1) entity-noun ambiguity errors, (2) entity boundary detection errors, and (3) entity-entity ambiguity errors. Due to these three problems, the gazetteer generating module presented in Section 2.1 is not adequate, by itself, for reliable named-entity recognition. We found heuristics in the literature to tackle each of these problems.

2.2.1 Entity-Noun Ambiguity

Entity-noun ambiguity occurs when an entity is the homograph of a noun. The plural word "jobs" and the surname "Jobs" is an example of this problem. To avoid this problem, Mikheev [9] proposes the following heuristic: In a given document, assume that a word or phrase with initial capitals (e.g., "Jobs") is a named-entity, unless (1) it sometimes appears in the document without initial capitals (e.g., "jobs"), (2) it only appears at the start of a sentence or at the start of a quotation (e.g., "Jobs that pay well are often boring."), or (3) it only appears inside a sentence in which all words with more than three characters start with a capital letter (e.g., a title or section heading).

2.2.2 Entity Boundary Detection

A common problem with the list lookup strategy is errors in recognizing where a named-entity begins and ends in a document (e.g., finding only "Boston" in "Boston White Sox"). This can happen when a named-entity is composed of two or more words (e.g., "Jean Smith") that are each listed separately (e.g., "Jean" as a first name and "Smith" as a last name). It can also happen when an entity is surrounded by unknown capitalized words (e.g., "New York Times" as an organization followed by "News Service" as an unlisted string). Palmer and Day [12] propose the longest match strategy for these cases. Accordingly, we merge all consecutive entities of the same type and every entity with any adjacent capitalized words. We did not, however, merge consecutive entities of different types, since we would not have known the resulting type.

The rule above is general enough to be applied independently of the entity type. We found that other merging rules could improve the precision of our system, such as "create a new entity of type organization by merging a location followed by an organization". However, we avoided rules like this, because we believe that this type of manual rule engineering results in brittle, fragile systems that do not generalize well to new data. Our goal is to make a robust, portable, general-purpose NER system, with minimal embedded domain knowledge.

2.2.3 Entity-Entity Ambiguity

Entity-entity ambiguity occurs when the string standing for a named-entity belongs to more than one type. For instance, if a document contains the named-entity “France”, it could be either the name of a person or the name of a country. For this problem, Petasis *et al.* [13], among others, propose that at least one occurrence of the named-entity should appear in a context where the correct type is clearly evident. For example, in the context “Dr. France”, it is clear that “France” is the name of a person.

We could have used cues, such as professional titles (e.g., farmer), organizational designators (e.g., Corp.), personal prefixes (e.g., Mr.) and personal suffixes (e.g., Jr.), but as discussed in the preceding section, we avoided this kind of manual rule engineering.

Definitions:

D = a given input document.

$A = \{a_1,...,a_n\}$ = the set of all sets of aliases in the document D.

$a_i = \{e_1,...,e_m\}$ = a set of aliases = a set of different entity instances, referring to the same actual entity in the world.

$e = \langle D, s, p \rangle$ = a unique instance of a named-entity, consisting of a string s in document D at position p.

$\text{overlap}(e_i, e_j)$ = a Boolean function; returns **true** when $e_i = \langle D, s_i, p_i \rangle$ and $e_j = \langle D, s_j, p_j \rangle$ and the strings s_i and s_j share at least one word with more than three characters; returns **false** otherwise.

Algorithm:

```
Let $A = \{\}$.
For each instance of a named-entity $e$ in document $D$:
    If there is exactly one alias set $a_i$ with a member $e_j$ such that
        $\text{overlap}(e, e_j)$, then modify $A$ by adding $e$ to $a_i$.
    If there are two or more alias sets $a_i$, $a_j$ with members $e_k$, $e_l$ such that
        $\text{overlap}(e, e_k)$ and $\text{overlap}(e, e_l)$, then modify $A$ by creating a new
        alias group $a_p$ that is the union of $a_i$, $a_j$, and $\{e\}$, add $a_p$ to $A$, and
        remove $a_i$ and $a_j$ from $A$.
    Otherwise, create a new alias set $a_q$, consisting of $\{e\}$, and add $a_q$ to $A$.
```

Fig. 1. Simple alias resolution algorithm

Instead, we applied a simple alias resolution algorithm, presented in Figure 1. When an ambiguous entity is found, its aliases are used in two ways. First, if a member of an alias set is unambiguous, it can be used to resolve the whole set. For instance, “Atlantic ocean” is clearly a location but “Atlantic” can be either a location or an organization. If both belong to the same alias set, then we assume that the whole set is of type *location*. A second way to use the alias resolution is to include unknown words in the model. Unknown words are typically introduced by the heuristic in Section 2.2.2. If an entity (e.g., “Steve Hill”) is formed from a known entity (e.g., “Steve”) and an unknown word (e.g., “Hill”), we allow occurrences of this unknown word to be added in the alias group.

3 Evaluation with the MUC-7 Enamex Corpus

In the Message Understanding Conferences (MUC), the Named-Entity Recognition (NER) track focuses on the three classical types of named-entities: person, location, and organization. These three types of named-entities are collectively called *Enamex*. In this section, we compare the performance of our system with a baseline supervised system, using the Enamex corpus from MUC-7. For this experiment, a portion of the corpus is given to the supervised system in order to train it. Our unsupervised system simply ignores this portion of corpus.

The same baseline experiment was conducted on MUC-6 and MUC-7 by Palmer and Day [12] and Mikheev *et al.* [10] respectively. Their systems work as follows. A training corpus is read and the tagged entities are extracted and listed. Given a testing corpus, the lists are used in a simple lookup strategy, so that any string that matches a list entry is classified accordingly.

Table 1 presents the results of Mikheev on MUC-7 (in the “Learned lists” columns). There is also a comparison with a system that uses hand-made lists of common entities (in the “Common lists” columns). The “Combined lists” columns are based on a combination of both approaches. These results are from Mikheev’s published experiments [10].

In Table 1, “re” is the recall, “pr” is the precision, and “f” is the f-measure (the harmonic mean of precision and recall), expressed as percentages.

Table 1. Results of a supervised system on MUC-7

	Learned lists			Common lists			Combined lists		
	re	Pr	f	re	pr	f	re	pr	f
organization	49	75	59	3	51	6	50	72	59
person	26	92	41	31	81	45	47	85	61
location	76	93	84	74	94	83	86	90	88

For the purpose of comparison, we ran our system on MUC-7 using gazetteers that we generated as described in Section 2.1. We generated gazetteers for some of the subtypes of named-entities given by Sekine [15]. The generated gazetteers are described in Table 2. We also used a special list of the months of the year, because we noticed they were an abnormally important source of noise on the development (dry run) set.[1] Many months are also valid as personal first names.

Table 2. Type and size of gazetteers built using Web page wrapper

Gazetteer	Size
Location: city	14,977
Location: state / province	1,587
Location: continent / country / island	781
Location: waterform	541
Location: astral body	85
Organization: private companies	20,498
Organization: public services	364
Organization: schools	3,387
Person: first names	35,102
Person: last names	3,175
Person: full names	3,791
Counter-examples: months	12

List size depends on the performance of the Web page wrapper at extracting entities. Nadeau [11] showed that lists have a precision of at least 90%. We did not restrict the web mining to a specific geographic region and we did not enforce strict conditions for the list elements. As a result, the "state / province" list contains elements from around the world (not only Canada and the U.S.) and the "first name" list contains a multitude of compound first names, although our algorithm is designed to capture them by merging sequences of first names, as explained in Section 2.2.2.

Table 3 shows the result of a pure list lookup strategy, based on our generated gazetteers (in the "Generated lists" columns). For comparison, Table 3 also shows the best supervised results from Table 1 (in the "Mikheev combined lists" columns). The results we report in Tables 1, 3, 4, and 5 are all based on the held-out formal corpus of MUC-7.

Table 3. Supervised list creation vs. unsupervised list creation techniques

	Mikheev combined lists			Generated lists		
	Re	pr	f	re	pr	f
organization	50	72	59	70	52	60
person	47	85	61	59	20	30
location	86	90	88	83	31	45

[1] It can be argued that the month list is a form of manual rule engineering, contrary to the principles discussed in Section 2.2.2. We decided to use it because most of the noise was clearly corpus-dependent, since each article contains a date header. For results without the month list, subtract 5% from the precision for the person type.

We believe the comparison in Table 3 gives a good sense of the characteristics of both approaches. The supervised approach is quite precise but its recall is lower, since it cannot handle rare entities. The unsupervised approach benefits from large gazetteers, which enable higher recall at the cost of lower precision.

The case of locations is interesting. There is evidence that there is a substantial vocabulary transfer between the training data and the testing data, which allows the supervised method to have an excellent recall on the unseen texts. Mikheev's lists get a high recall with a list of only 770 locations. The supervised method benefits from highly repetitive location names in the MUC corpus.

These results are slightly misleading. The MUC scoring software that produces these measures allows partial matching. That means, if a system tags the expression "Virgin Atlantic" when the official annotated key is "Virgin Atlantic Group", it will be credited with a success. In Table 4, we provide another view of the system's performance, which may be less misleading. Table 4 gives, for our system, the precision and recall of all entity types at the level of *text*; that is, the performance on finding exact string matches.

Table 4. Generated list performance on text matching

	Generated lists		
	re	pr	f
text	61	29	39

The next step in our evaluation consists in adding the heuristics presented in Sections 2.2.1 to 2.2.3. These heuristics are designed to be unsupervised; that is, they require no training (unlike n-gram contexts, for example) and they are not deduced from our domain knowledge about a specific entity type. Table 5 shows the contribution of each heuristic. The "Generated lists" columns are copied from Tables 3 and 4, to show the performance of the list lookup strategy without disambiguation (i.e., Section 2.1 without Section 2.2).

Table 5. Performance of heuristics to resolve named-entity ambiguity

	Generated lists			H1 (Entity-noun ambiguity)			H1 + H2 (Entity boundary)			H1 + H2 + H3 (Entity-entity ambiguity)		
	re	pr	f	re	pr	f	re	pr	f	re	pr	f
org.	70	52	60	69	73	71	69	74	71	71	75	73
per.	59	20	30	58	53	55	66	63	64	83	71	77
loc.	83	31	45	82	69	75	81	77	79	80	77	78
text	61	29	39	61	57	59	72	72	72	74	72	73

The contribution of each heuristic (H1, H2, H3) is additive. H1 (Section 2.2.1) procures a dramatic improvement in precision with negligible loss of recall. The main source of ambiguity is entity-noun homographs such as "jobs", "gates", and "bush".

Heuristic H2 (Section 2.2.2) gives small gains in precision and recall of individual entity types (the first three rows in Table 5). As explained, these scores are misleading because they count partial matches and thus these scores are not sensitive to the boundary detection errors that are corrected by H2. However, the performance of text matching is greatly improved (last row in Table 5). We noticed that most corrected boundaries are attributable to person entities composed of a known first name and an unlisted capitalized string standing, presumably, for the surname.

H3 (Section 2.2.3) mainly increases precision and recall for named-entities of the person type, due to the the alias resolution algorithm. An occurence of a full person name is usually unambiguous and thus can help with annotating isolated surnames, which are often either ambiguous (confused with organization names) or simply unlisted strings.

4 Evaluation with Car Brands

There are many more types of named-entities than the three classical types in Enamex. Sekine *et al.* [15] propose a hierarchy of 200 types of named-entities. Evans [5] proposes a framework to handle such wide variety. His approach is based on lexical patterns, inspired by Hearst [6]. He paired this technique with a heuristic for handling ambiguity in capitalized words. Our system is similar, but it is based on a method proven to give better recall at finding entities [4].

In this section, we show how the system performs on the task of recognizing car brands. Intuitively, it seems that this type is easier to handle than a type such as persons that has an almost infinite extension. However, recognizing car brands poses many difficulties. Car brands can be confused with common nouns (e.g., Focus, Rendez-Vous, Matrix, Aviator) and with company names (e.g., "Ford" versus "Ford Motor Company"). Another difficulty is the fact that new car brands are created every year, so keeping a gazetteer of car brands up-to-date is challenging.

We created a small pilot corpus composed of news specifically about cars from some popular news feeds (CanWest, National Post, and The Associated Press). We use eight documents, for a total of 5,570 words and 196 occurrences of car brands.

The Web-page wrapper technique was used to generate a list of 5,701 car brands and the heuristics of sections 2.2.1 to 2.2.3 were applied without any modifications. Table 6 reports the results.

Table 6. System performance for car brand recognition

	Generated list			H1, H2 and H3		
	Re	pr	f	re	pr	f
cars	86	42	56	85	88	86
text	71	34	46	79	83	81

The performance on this task is comparable to the Enamex task. Without ambiguity resolution (in the "Generated list" columns), the precision is low, typically under 50%. This is the impact of frequent and ambiguous words like "will" (Toyota Will) and noise in our list (e.g., new, car, fuel). The ambiguity resolution algorithms (in the "H1, H2, and H3" columns) raise the precision above 80%. The remaining recall errors are due to rare car brands (e.g., "BMW X5 4.8is" or "Ford Edge"). The remaining precision errors are due to organization-car ambiguity (e.g., "National" as in "National Post" versus "Chevrolet National") and noise in the list (e.g., Other, SUV). We believe that the good performance of gazetteer generation combined with ambiguity resolution on an entirely new domain emphasizes their domain-independent character and shows the strength of the unsupervised approach.

5 Supervised versus Unsupervised

We describe our system as unsupervised, but the distinction between supervised and unsupervised systems is not always clear. In some systems that are apparently unsupervised, it could be argued that the human labour of generating labeled training data has merely been shifted to embedding clever rules and heuristics in the system.

In our gazetteer generator (Section 2.1), the supervision is limited to a seed of four entities per list Less than four examples results in lower precision and more than four examples results in lower recall [4]. In our ambiguity resolver (Section 2.2), we attempt to minimize the use of domain knowledge of specific entity types. Our system exploits human-generated HTML markup in Web pages to generate gazetteers. However, because Web pages are available in such a quantity and because the creation of Web pages is now intrinsic to the workflow of most organization and individuals, we believe this annotated data comes at a negligible cost. For these reasons, we believe it is reasonable to describe our system as unsupervised.

6 Conclusion

In this paper, we presented a named-entity recognition system that advances the state-of-the-art of NER by avoiding the need for supervision and by handling novel types of named-entities. In a comparison on the MUC corpus, our system outperforms a baseline supervised system but it is still not competitive with more complex supervised systems. There are (fortunately) many ways to improve our model. One interesting way would be to generate gazetteers for a multitude of named-entity types (e.g., all 200 of Sekine's types) and use list intersection as an indicator of ambiguity. This idea would not resolve the ambiguity itself but would clearly identify where to invest further efforts.

Acknowledgements

We would like to thank Caroline Barrière, who provided us with helpful comments on an earlier version of this work. Support of the Natural Sciences and Engineering Research Council and of the Communications and Information Technology division of the Ontario Centres of Excellence is gratefully acknowledged.

References

1. Chinchor, N. (1998) MUC-7 Named Entity Task Definition, version 3.5. *Proc. of the Seventh Message Understanding Conference*.
2. Cohen, W. and Fan, W. (1999) Learning Page-Independent Heuristics for Extracting Data from Web Page, *Proc. of the International World Wide Web Conference*.
3. Collins M. and Singer, Y. (1999) Unsupervised Models for Named Entity Classification. *Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.
4. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S. and Yates, A. (2005) Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 165, pp. 91-134.
5. Evans, R. (2003) A Framework for Named Entity Recognition in the Open Domain. *Proc. Recent Advances in Natural Language Processing*.
6. Hearst, M. (1992) Automatic Acquisition of Hyponyms from Large Text Corpora. *Proc. of International Conference on Computational Linguistics*.
7. Lin, D. and Pantel, P. (2001) Induction of Semantic Classes from Natural Language Text. *Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
8. Ling, C., and Li, C. (1998). Data Mining for Direct Marketing: Problems and Solutions. *Proc. International Conference on Knowledge Discovery and Data Mining.*
9. Mikheev, A. (1999) A Knowledge-free Method for Capitalized Word Disambiguation. *Proc. Conference of Association for Computational Linguistics*.
10. Mikheev, A., Moens, M. and Grover, C. (1999) Named Entity Recognition without Gazetteers. *Proc. Conference of European Chapter of the Association for Computational Linguistics*.
11. Nadeau, D. (2005) Création de surcouche de documents hypertextes et traitement du langage naturel, Proc. *Computational Linguistics in the North-East*.
12. Palmer, D. D. and Day, D. S. (1997) A Statistical Profile of the Named Entity Task. *Proc. ACL Conference for Applied Natural Language Processing*.
13. Petasis, G., Vichot, F., Wolinski, F., Paliouras, G., Karkaletsis, V. and Spyropoulos, C. D. (2001) Using Machine Learning to Maintain Rule-based Named-Entity Recognition and Classification Systems. *Proc. Conference of Association for Computational Linguistics*.
14. Riloff, E. and Jones, R (1999) Learning Dictionaries for Information Extraction using Multi-level Bootstrapping. *Proc. of National Conference on Artificial Intelligence*.
15. Sekine, S., Sudo, K., Nobata, C. (2002) Extended Named Entity Hierarchy, *Proc. of the Language Resource and Evaluation Conference*.
16. Zhu, X., Wu, X. and Chen Q. (2003) Eliminating Class Noise in Large Data-Sets, *Proc. of the International Conference on Machine Learning*.