Dependency-Based Open Information Extraction

Pablo Gamallo and **Marcos Garcia**
Centro de Investigação sobre Tecnologias da Informação (CITIUS)
Universidade de Santiago de Compostela
pablo.gamallo@usc.es marcos.garcia.gonzalez@usc.es

Santiago Fernández-Lanza
Escola Superior de Enxeñería Informática
Universidade de Vigo
sflanzal@uvigo

Abstract

Building shallow semantic representations from text corpora is the first step to perform more complex tasks such as text entailment, enrichment of knowledge bases, or question answering. Open Information Extraction (OIE) is a recent unsupervised strategy to extract billions of basic assertions from massive corpora, which can be considered as being a shallow semantic representation of those corpora. In this paper, we propose a new multilingual OIE system based on robust and fast rule-based dependency parsing. It permits to extract more precise assertions (verb-based triples) from text than state of the art OIE systems, keeping a crucial property of those systems: scaling to Web-size document collections.

1 Introduction

There is an increasing interest in capturing shallow semantic representations from large amounts of text, with the aim of elaborating more complex semantic tasks involved in text understanding, such as textual entailment, filling knowledge gaps in text, or integration of text information into background knowledge bases. Two recent approaches to text understanding are interested in shallow semantics: Machine Reading (Etzioni et al., 2006) and Learning by Reading (Barker et al., 2007). Both approaches aim at understanding text by starting with a very basic representation of the facts conveyed by the input text. In addition, they rely on unsupervised strategies. There are, however, two significant differences between Machine Reading and Learning by Reading:

The first difference concerns the basic representation required at the beginning of the understanding process. While Machine Reading is focused on fixed structures (triples), constituted by a relation (a verb or verb phrase) and two arguments, in Learning by Reading the text is represented by means of more flexible predicate-argument structures (n-tuples) derived from syntactic dependency trees. In Learning by Reading, on the one hand, relations with more than two arguments are also extracted, and on the other, relations are not restricted to verb phrases but to whatever relation expressed by a dependency based triple, *(head, relation, modifier)*, also called Basic Element (Hovy et al., 2005). The second difference is related to the notion of text domain. Whereas Machine Reading works on open relations and unrestricted topics and domains, Learning by Reading prefers being focused on domain-specific texts in order to build a semantic model of a particular topic.

One of the major contributions of Machine Reading is the development of an extraction paradigm, called Open Information Extraction (OIE), which aims at extracting a large set of verb-based triples (or assertions) from unrestricted text. An OIE system reads in sentences and rapidly extracts one or more textual assertions, consisting in a verb relation and two arguments, which try to capture the main relationships in each sentence (Banko et al., 2007). Unlike most relation extraction methods which are focused on a predefined set of target relations, OIE is not limited to a small set of target relations known in advance, but extracts all types of (verbal) binary relations found in the text. The OIE system with best performance, called ReVerb (Etzioni et al., 2011), is a logistic regression classifier that takes as input PoS-tagged and NP-chunked sentences. So,

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 10–18,
Avignon, France, April 23 - 27 2012. ©2012 Association for Computational Linguistics

it only requires shallow syntactic features to generate semantic relations, guaranteeing robustness and scalability with the size of the corpus. One of the main critics within the OIE paradigm against dependency based methods, such as Learning by Reading, concerns the computational cost associated with rich syntactic features. Dependency parsing could improve precision and recall over shallow syntactic features, but at the cost of extraction speed (Etzioni et al., 2011). In order to operate at the Web scale, OIE systems needs to be very fast and efficient.

In this paper, we describe an OIE method to generate verb-based triples by taking into account the positive properties of the two traditions: considering Machine Reading requirements, our system is efficient and fast guaranteeing scalability as the corpus grows. And considering ideas behind Learning by Reading, we use a dependency parser in order to obtain fine-grained information (e.g., internal heads and dependents) on the arguments and relations extracted from the text. In addition, we make extraction multilingual. More precisely, our system has the following properties:

- Unsupervised extraction of triples represented at different levels of granularity: surface forms and dependency level.

- Multilingual extraction (English, Spanish, Portuguese, and Galician) by making use of a multilingual rule-based parser, called DepPattern (Gamallo and González, 2011).

Our claim is that it is possible to perform Open Information Extraction by making use of very conventional tools, namely rule-based dependency analysis and simple post-processing extraction rules. In addition, we also show that we can deal with knowledge-rich syntactic information while remaining scalable.

This article is organized as follows. Section 2 introduces previous work on OIE: in particular it describes three of the best known OIE systems up to date. Next, in Section 3, the proposed method is described in detail. Then, some experiments are performed in Section 4, where our OIE system is compared against ReVerb. In 5, we sketch some applications that use the output of our OIE system, and finally, conclusions and current work are addressed in 6.

2 Open Information Extraction Systems

An OIE system extracts a large number of triples *(Arg1, Rel, Arg2)* for any binary relation found in the text. For instance, given the sentence "Vigo is the largest city in Galicia and is located in the northwest of Spain", an OIE system should extract two triples: *(Vigo, is the largest city in, Galicia)* and *(Vigo, is located in, northwest of Spain)*. Up to now, OIE is focused only on verb-based relations. Several OIE systems have been proposed, all of them are based on an extractor learned from labelled sentences. Some of these systems are:

- TextRunner (Banko et al., 2008): the extractor is a second order linear-chain CRF trained on samples of triples generated from the Penn Treebank. The input of TextRunner are PoS-tagged and NP-chunked sentences, both processes performed with OpenNLP tools.

- WOE (Wu and Weld, 2010): the extractor was learned by identifying the shortest dependency paths between two noun phrases, using training examples of Wikipedia. The main drawback is that extraction is 30 times slower than TextRunner.

- ReVerb (Etzioni et al., 2011; Fader et al., 2011): the extractor is a logistic regression classifier trained with shallow syntactic features, which also incorporates lexical constraints to filter out over-specified relation phrases. It takes as input the same features as TextRunner, i.e., PoS-tagged and NP-chunked sentences analyzed with OpenNLP tools. It is considered to be the best OIE system up to now. Its performance is 30% higher than WOE and more than twice that of TextRunner.

One of the most discussed problems of OIE systems is that about 90% of the extracted triples are not concrete facts (Banko et al., 2007) expressing valid information about one or two named entities, e.g. "Obama was born in Honolulu". However, the vast amount of high confident relational triples extracted by OIE systems are a very useful startpoint for further NLP tasks and applications, such as common sense knowledge acquisition (Lin et al., 2010), and extraction of domain-specific relations (Soderland et al., 2010). The objective of OIE systems is not to extract concrete facts, but to transform unstructured texts into structured information, closer to ontology formats.

Nevertheless, some linguistics problems arise. OIE systems were trained to identify only verb clauses within the sentences and, therefore, to extract just binary verb-based relations from the clause structure. It follows that they cannot be easily adapted to learn other non-clausal relations also found in the text. Let us take the following sentence: "The soccer player of FC Barcelona, Lionel Messi, won the Fifa World Player of the Year award". In addition to the main verb-based relationship:

> *(Lionel Messi, won, the Fifa Worlds Player of the Year award)*

which could be extracted by the OIE systems introduced above, it should also be important to extract other non-verbal relations found within the noun phrases:

(Messi, is, a soccer player of FC Barcelona)
(Fifa World Player of the Year, is, an award)

However, the cited systems were not trained to learn such a basic relations.

Besides, the OIE systems are not adapted to process clauses denoting events with many arguments. Take the sentence: "The first commercial airline flight was from St. Petersburg to Tampa in 1914". We should extract, at least, two or three different relational triples from the verb clause contained in this sentence, for instance:

(the first commercial airline flight, was from, St. Petersburg)
(the first commercial airline flight, was to, Tampa)
(the first commercial airline flight, was in, 1914)

Yet, current OIE systems are not able to perform this multiple extraction. Even if the cited OIE systems can identify several clauses per sentence, they were trained to only extract one triple per clause.

In the following, we will describe a dependency-based OIE system that overcomes these linguistic limitations.

3 A Dependency-Based Method for Open Information Extraction

The proposed extraction method consists of three steps organized as a chain of commands in a pipeline:

Dependency parsing Each sentence of the input text is analyzed using the dependency-based parser DepPattern, a multilingual tool available under GPL license[1].

Clause constituents For each parsed sentence, we discover the verb clauses it contains and, then, for each clause, we identify the verb participants, including their functions: subject, direct object, attribute, and prepositional complements.

Extraction rules A set of rules is applied on the clause constituents in order to extract the target triples.

These three steps are described in detail below.

3.1 Dependency Parsing

To parse text, we use an open-source suite of multilingual syntactic analysis, DepPattern (Gamallo and González, 2011). The suite includes basic grammars for five languages as well as a compiler to build parsers in Perl. A parser takes as input the output of a PoS-tagger, either, FreeLing (Carreras et al., 2004) or Tree-Tagger[2]. The whole process is robust and fast. It takes 2600 words per second on a Linux platform with 2.4GHz CPU and 2G memory. The basic grammars of DepPattern contain rules for many types of linguistic phenomena, from noun modification to more complex structures such as apposition or coordination. However their coverage is still not very high. We added several rules to the DepPattern grammars in English, Spanish, Portuguese, and Galician, in order to improve the coverage of our OIE system.

The output of a DepPattern parser consists of sentences represented as binary dependencies from the head lemma to the dependent lemma: *rel(head, dep)*. Consider the sentence "The coach of Benfica has held a press conference in Lisbon".

[1] `htpp://gramatica.usc.es/pln/tools/deppattern.htm`

[2] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html`

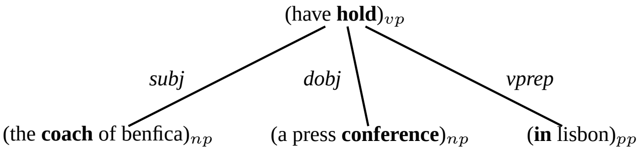


Figure 1: Constituency tree with function information

The DepPattern dependencies are the following:

spec(coach-2, the-1)
nprep(coach-2, of-3)
term(of-3, benfica-4)
aux(hold-6, have-5)
subj(hold-6, coach-2)
dobj(hold-6, conference-9)
spec(conference-9, a-7)
modif(conference-9, press-8)
vprep(hold-6, in-10)
term(in-10, lisbon-11)

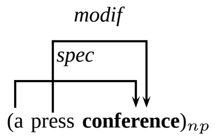
The directed graph formed by these dependencies will be the input of the following step.

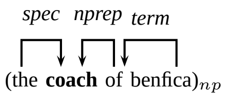
3.2 Clause Constituents

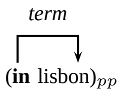
In the second step, we identify the clauses of each sentence, and, for each clause, we retain the participants and their functions with regard to the verb of the clause. A sentence can contain several clauses, in particular, we identify the main clause, relative clauses, and that-clauses.

In our example, there is just one clause constituted by a verb phrase (*"have hold"*) and three participants: the subject *"the coach of benfica"*, the direct object *"a press conference"*, and a prepositional phrase *"in lisbon"*. So, the objective here is to transform the dependency path built in the first step into a partial constituency tree, where only the constituents of the clause are selected. The process of constructing the clause constituents and the verb phrase is as follows.

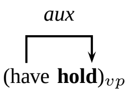
Given a verb dependency (namely *subj*, *dobj*, *vprep*, or *attrib*), we select the dependent lemma of the clause verb and then we list all dependent lemmas linked to the target lemma (as a head) through the syntactic dependency path. It results in the construction of the main phrases of the clause, including information about the head of the phrase. We show below the three constituents identified from our example, where the directed arrows stand for the internal dependencies used for their identification (the head of each phrase is in bold):







The verb phrase is also built in a similar way. It contains all dependent lemmas of the verb that are not part of the clause constituents identified before:



The three clause constituents are also provided with information about their function with regard to the clause verb, as Figure 1 shows. The function of a constituent inherits the name of the dependent relation linking the clause verb to the head of the constituent. For instance, the function of (the **coach** of benfica)$_{np}$ is the name of the dependent relation in *subj(hold-6, coach-2),* that is *subj*. The clause constituents as well as the verb phrase of each clause are the input of the extraction rules.

3.3 Extraction Rules

The third and last process consists of a small set of simple extraction rules that are applied on the clauses identified in the previous step. The output of an extraction rule is a triple whose internal word tokens are provided with some linguistic information: lemma, PoS tag, head of the constituent, etc.

The simplest rule is applied on a clause just containing a subject and a direct object. In such a case, the two constituents are the arguments of the triple, while the verb phrase is the relation.

In our previous example, the clause contains three arguments: a subject (*"the coach of benfica"*), a direct object (*"a press conference"*), and a prepositional complement (*"in Lisbon"*). In this case, our strategy is similar to that of ReVerb system, namely to consider the relation as the verb phrase followed by a noun phrase and ending in a preposition. For this purpose, we have defined an extraction rule that builds the relation of the triple using the verb phrase, the direct object, and the head preposition of the prepositional phrase: *"have hold a press conference in"*. The two arguments are: *"the coach of benfica"* and *"Lisbon"*. The triple generated by our rule is represented as follows:

ARG1:	*the_DT coach_N-H of_PRP benfica_N*
REL:	*have_V hold_V-H a_DT press_N conference_N-H in_PRP*
ARG2:	*Lisbon_N-H*

which contains lemmas, PoS tags (DT, N, PRP,...), as well as the heads (tag "H") of the main constituents. In addition to this syntax-based representation, the extraction rule also gives us a surface form of the triple with just tokens:

(the coach of Benfica, has hold a press conference in, Lisbon)

Table 1 shows the main rules we defined to extract triples from patterns of clause arguments. The order of arguments within a pattern is not relevant. The argument 'vprep' stands for a prepositional complement of the verb, which consists of a preposition and a nominal phrase (np). The third row represents the extraction rule used in our previous example. All rules in Table 1 are applied at different clause levels: main clauses, relative clauses and that-clauses.

As in the case of all current OIE systems, our small set of rules only considers verb-based clause triples and only extract one triple per clause. We took this decision in order to make a fair comparison when evaluating the performance of our system against ReVerb (in the next section). However, nothing prevents us from writing extraction rules to generate several triples from one clause with many arguments, or to extract triples from other patterns of constituents, for instance: vp-pp-pp, noun-prep-noun, noun-noun, adj-noun, or verb-adverb..

patterns	**triples**

subj-vp-dobj	Arg1 = subj
	Rel= vp
	Arg2 = dobj

subj-vp-vprep	Arg1 = subj
	Rel= vp+prep (prep from vprep)
	Arg2 = np (from vprep)

subj-vp-dobj-vprep	Arg1 = subj
	Rel= vp+dobj+prep
	Arg2 = np (from vprep)

subj-vp-attr	Arg1 = subj
	Rel= vp
	Arg2 = attr

subj-vp-attr-vprep	Arg1 = subj
	Rel= vp+attr+prep (from vprep)
	Arg2 = np (from vprep)

Table 1: Pattern based rules to generate final triples

Finally, let us note that current OIE systems, such as ReVerb, produces triples only in textual, surface form. Substantial postprocessing is needed to derive relevant linguistic information from the tuples. By contrast, in addition to surface form triples, we also provide syntax-based information, PoS tags, lemmas, and heads. If more information is required, it can be easily obtained from the dependency analysis.

4 Experiments

4.1 Wikipedia Extraction

The system proposed in this paper, hereafter DepOE, was used to extract triples from the Wikipedia in four languages: Portuguese, Spanish, Galician, and English.[3] Before applying the extractor, the xml files containing the Wikipedia were properly converted into plaintext. The number of both sentences and extracted triples are shown in Table 2. We used PoS-tagged text with Tree-Tagger as input of DepPattern for the English extraction, and FreeLing for the other three languages. Note that, unlike OIE systems described in previous work, DepOE can be considered as being a multilingual OIE system.[4]

[3] Wikipedia dump files were downloaded at `http://download.wikipedia.org` on September 2010.

[4] DepOE is an open source system freely available, under GPL license, at `http://gramatica.usc.es/~gamallo/prototypes.htm`.

Wikipedia version	sentences	triples
English	$78,826,696$	$47,284,799$
Spanish	$21,208,089$	$6,527,195$
Portuguese	$11,714,672$	$3,738,922$
Galician	$1,461,705$	$480,138$

Table 2: Number of sentences and triples from four Wikipedias

It is worth mentioning that the number of extracted triples is lower than that obtained with ReVerb, which reaches $63,846,865$ triples (without considering a threshold for confidence scores). This is due to the fact that the DepPattern grammars are not complete and, then, they do not perform deep analysis, just partial parsing. In particular, they do not consider all types of coordination and do not deal with significant linguistic clausal phenomena such as interrogative, conditional, causal, or adversative clauses. Preliminary evaluations of the four parsers showed that they behave in a similar way, yet Portuguese and Galician parsers achieve the best performance, about 70% f-score.

In this paper, we do not report experimental evaluation of the OIE system for languages other than English.

4.2 Evaluation

We compare Dep-OE to ReVerb[5], regarding the quantity and quality of extracted triples just in English, since ReVerb only can be applied on this language. Each system is given a set of sentences as input, and returns a set of triples as output. A test set of 200 sentences was created by randomly selecting sentences from the English Wikipedia. Each test sentence was independently examined by two judges in order to, on the one hand, identify the triples actually contained in the sentence, and on the other, evaluate each extraction as correct or incorrect. Incoherent and uninformative extractions were considered as incorrect. Given the sentence “The relationship between the Taliban and Bin Laden was close”, an example of incoherent extraction is:

(Bin Laden, was, close)

Uninformative extractions occur when critical information is omitted, for instance, when one of the arguments is truncated. Given the sentence “FBI examined the relationship between Bin Laden and the Taliban”, an OIE system could return a truncated triple:

(FBI, examined the relationship between, Bin Landen)

We follow similar criteria to those defined in previous OIE evaluations (Etzioni et al., 2011).

Concerning the decisions taken by the judges on the extractions made by the systems, the judges reached a very high agreement, 93%, with an agreement score of $\kappa = 0.83$. They also reached a high agreement, 86%, with regard to the number of triples (gold standard) found in the test sentences.

The precision of a system is the number of extractions returned as correct by the system divided by the number of returned extractions. Recall is the number of extractions returned as correct by the system divided by the number of triples identified by the judges (i.e., the size of the gold standard). Moreover, to compare our rule-based system DepOE to ReVerb, we had to select a particular threshold restricting the extractions made by ReVerb. Let us note that this extractor is a logistic regression classifier that assign confidence scores to its extractions. We computed precision and recall for many threshold and selected that giving rise to the best f-score. Such a threshold was 0.15. So, we compare DepOE to the results given by ReVerb for those extractions whose confidence score is higher than 0.15.

As it was done in previous OIE evaluations, the judges evaluated two different aspects of the extraction:

- how well the system identify correct relation phrases,
- the full extraction task, i.e., whether the system identifies correct triples (both the relation and its arguments).

Figures 2 and 3 represent the score average obtained by the two judges. They show that DepOE system is more precise than ReVerb. This is clear in the full extraction task, where DepOE achieves 68% precision while ReVerb reaches 52%. By contrast, as it was expected, DepOE has lower recall because of the low coverage of the grammars it depends on. Regarding f-score, DepOE

[5] `http://reverb.cs.washington.edu/`

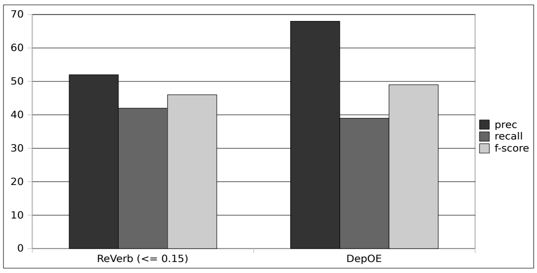


Figure 2: Evaluation of the extraction of triples (both relation and its arguments) performed by DepOE and ReVerb (with a confidence score $>= 0.15$).

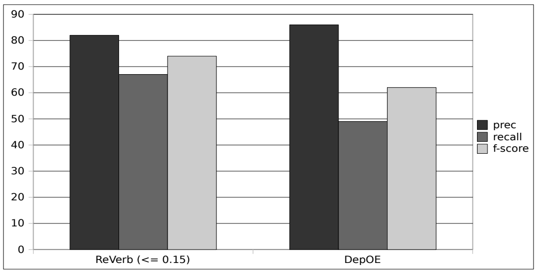


Figure 3: Evaluation of the relation extraction performed by DepOE and ReVerb (with a confidence score $>= 0.15$).

performs better than ReVerb in the full extraction task, but when only relations are considered, ReVerb achieves the highest score.

We found that most of the incorrect extractions returned by the two systems where cases where the relation phrase was correctly identified, but not one of the arguments. However, there are significant differences between the two systems concerning the type of problems arising in argument identification.

The most common errors of ReVerb are both: incorrect identification of the first argument (arg1) and extraction of only a truncated part of the second argument (arg2), as in the case of coordinating conjunctions. These two problems are crucial for ReVerb since more than 60% of incorrect extractions were cases with incorrect arguments and correct relations. DepOE has more precise extractions of the two arguments, in particular of arg1, since the parser is able to correctly identify the subject. Nevertheless, it also produces many truncated arg2. Let us see an example. Given the sentence "Cities and towns in Romania can have the status either of municipiu or oras", ReVerb was not able to identify the correct arg1 and returned a truncated arg2:

(Romania, can have, the status)

DepOE correctly identified the subject (arg1) but also failed to return the correct arg2:

(Cities and towns in Romania, can have, the status)

In general, when DepOE fails to correctly identify an argument, it is often trivial to find the reason of the problem. In the example above, arg2 was truncated because the English grammar has not any specific rule linking the particle "either" to a coordinate expression. So, the improvement of DepOE depends on improving the grammars it is based on. Besides the low coverage of the grammar, there are other sources of problems concerning the correct identification of arguments. In particular, it is worth mentioning that the English version of DepOE is not provided with an efficient Named Entity Recognition system. This makes it difficult to correctly identify multiword arguments with Named Entities, quantities, measures, and dates. Such a problem was partially solved by the use of FreeLing in the Portuguese, Spanish, and Galician DepOE versions.

4.3 Extraction Speed

To test the system's speed, we ran each extractor on the $100,000$ first lines of the English Wikipedia using a Linux platform with 2.4GHz CPU and 2GB memory. The processing time of ReVerb was 4 minutes while that of DepOE was 5 minutes and 19 seconds. In this platform, ReVerb is able to process $2,500$ words per second, and DepOE $1,650$. Concerning the use of RAM, ReVerb requires the 27% memory of the computer, while DepOE only needs 0.1%.

5 Applications

The extracted triples can be used for several NLP applications. The first application we are developing is a multilingual search engine over the triples extracted from the Wikipedia. All triples are indexed with Apache Solr[6], which enables it to rapidly answer queries regarding the extracted information, as in the query form of ReVerb[7].

Another application is to use the extracted triples to discover commonsense knowledge of

[6] http://lucene.apache.org/solr/

[7] http://textrunner.cs.washington.edu/reverb_demo.pl

team play game
team win championship
team win medal
team win game
team play match
organism have DNA
organism use energy
organism recycle detritus
organism respond to selection
organism modify environment

Table 3: Some of the most frequent basic propositions containing the words “team” and “organism”, discovered by our system from Wikipedia.

specific domains. One of the goals of Learning by Reading is to enable a computer to acquire basic knowledge of different domains in order to improve question answering systems (Hovy et al., 2011). We assume that the head expressions of the most frequent triples extracted from a specific domain represent basic propositions (common knowledge) of that domain.

To check this assumption, we built two domain-specific corpora from Wikipedia: a corpus constituted by articles about sports, and another corpus with articles about Biology. Then, we extracted the triples from those corpora and, for each triple, we selected just the head words of its three elements: namely the main verb (and preposition if any) of the relation and the head nouns of the two arguments. It resulted in a list of basic propositions of a specific domain. Table 3 shows some of the propositions acquired following this method. They are some of the most frequent propositions containing two specific words, “team” and “organism”, in the subject position (arg1) of the triples. The propositions with “team” were extracted from the corpus about sports, while those with “organism” were acquired from the corpus of Biology.

6 Conclusions and Current Work

We have described a multilingual Open Information Extraction method to extract verb-based triples from massive corpora. The method achieves better precision than state of the art systems, since it is based on deep syntactic information, namely dependency trees. In addition, given that dependency analysis is performed by fast, robust, and multilingual parsers, the method is scalable and applied to texts in several languages: we made experiments in English, Portuguese, Spanish, and Galician.

Our work shows that it is possible to perform Open Information Extraction by making use of knowledge-rich tools, namely rule-based dependency parsing and pattern-based extraction rules, while remaining scalable.

Even if in the experiments reported here we did not deal with relationships that are not binary, the use of deep syntactic information makes it easy to build n-ary relations from such cases, for instance complex events with internal (subject and object) and external (time and location) arguments: “*The treaty was signed by Portugal in 2003 in Lisbon*”. Furthermore, the use of deep syntactic information will also be useful to find important relationships that are not expressed by verbs. For instance, from the noun phrase *“Nobel Prize”*, we should extract the basic proposition: *(Nobel, is_a, prize)*.

In current work, we are working on synonymy resolution for two different cases found in the extracted triples: first, the case of multiple proper names for the same named entity and, second, the multiple ways a relationship can be expressed. Concerning the latter case, to solve relationship synonymy, we are making use of classic methods for relation extraction. Given a predefined set of target relations, a set of lexico-syntactic patterns is learned and used to identify those triples expressing the same relationship. This way, traditional closed information extraction could be perceived as a specific task aimed at normalizing and semantically organizing the results of open information extraction.

Acknowledgments

This work has been supported by the MICINN, within the projects with reference FFI2010-14986 and FFI2009-08828, as well as by *Diputación de Ourense* (INOU11A-04).

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *International Joint Conference on Artificial Intelligence*.

Michele Banko, , and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Annual Meeting of the Association for Computational Linguistics*.

K. Barker, B. Agashe, S. Chaw, J. Fan, N. Friedland, M. Glass, J. Hobbs, E. Hovy, D. Israel, D.S. Kim, et al. 2007. Learning by reading: A prototype system, performance baseline and lessons learned. In *Proceeding of Twenty-Second National Conference of Artificial Intelligence (AAAI 2007)*.

X. Carreras, I. Chao, L. Padró, and M. Padró. 2004. An Open-Source Suite of Language Analyzers. In *4th International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal.

Oren Etzioni, Michele Banko, and Michael J. Cafarella. 2006. Machine reading. In *AAAI Conference on Artificial Intelligence*.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: the second generation. In *International Joint Conference on Artificial Intelligence*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Conference on Empirical Methods in Natural Language Processing*.

Pablo Gamallo and Isaac González. 2011. A grammatical formalism based on patterns of part-of-speech tags. *International Journal of Corpus Linguistics*, 16(1):45–71.

Eduard Hovy, Chin yew Lin, and Liang Zhou. 2005. A BE-based Multi-document Summarizer with Sentence Compression. In *Proceedings of Multilingual Summarization Evaluation (ACL workshop). Ann Arbor, MI*.

Dirk Hovy, Chunliang Zhang, Eduard Hovy, and Anselmo Pe nas. 2011. Unsupervised discovery of domain-specific knowledge from text. In *Proceedings of 49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon, USA.

Thomas Lin, Mausman, and Oren Etzioni. 2010. Identifying functional relations in web text. In *Conference on Empirical Methods in Natural Language Processing*.

Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni. 2010. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Annual Meeting of the Association for Computational Linguistics*.