Linguistic Patterns for Information Extraction in OntoCmaps

Amal Zouaq[1,2], Dragan Gasevic[2,3], Marek Hatala[3]

[1]Royal Military College of Canada, CP 17000, Succ. Forces, Kingston, ON Canada K7K 7B4
[2]Athabasca University, 1 University Drive, Athabasca, AB T9S 3A3
[3]Simon Fraser University, 250-102nd Avenue, Surrey, BC Canada V3T 0A3
Amal.zouaq@rmc.ca, dgasevic@acm.org, mhatala@sfu.ca

Abstract. Linguistic patterns have proven their importance for the knowledge engineering field especially with the ever-increasing amount of available data. This is especially true for the Semantic Web, which relies on a formalization of knowledge into triples and linked data. This paper presents a number of syntactic patterns, based on dependency grammars, which output triples useful for the ontology learning task. Our experimental results show that these patterns are a good starting base for text mining initiatives in general and ontology learning in particular.

Keywords: Linguistic patterns, dependency grammars, triples, knowledge extraction, ontology learning.

1 Introduction

With the development of Semantic Web technologies and the increased number of initiatives relative to the Web of data, there is a need to create reusable and high quality ontologies. For this purpose, ontology design patterns (ODP) have been modeled and span various aspects of ontological design such as architectural ODPs, Content ODPs and Reengineering ODPs [1]. These patterns enable the definition of formal methodologies for ontology creation and maintenance. Of particular interest to the knowledge engineering community are Lexico-Syntactic Patterns. In fact, with the availability of large unstructured knowledge resources such as Wikipedia, it becomes crucial to be able to extract ontological content using scalable (semi)automatic knowledge extraction techniques. In this work, we consider that lexico-syntactic patterns are syntactic structures that trigger the extraction of chunks of information. Obviously, these patterns cannot be used in isolation and they necessitate various filtering mechanisms [6] to identify *ontological knowledge* from this extracted *textual information*. However, by constituting Lexico-syntactic ODP catalogs, it is likely that these ODPs will be used and reused as the first building block of ontology learning initiatives. So far, there are some ODPs repositories [2] but their number remains modest.

adfa, p. 1, 2011.

Trying to address this issue from a knowledge extraction perspective, this paper introduces the main patterns that are used by OntoCmaps, an ontology learning tool [6]. OntoCmaps exploits a pattern knowledge base that is domain independent. OntoCmaps lexico-syntactic ODPs are based on a dependency grammar formalism [5] that is well-suited for knowledge extraction. This paper describes the various ODPs that are used to extract multi-word expressions, hierarchical relationships and conceptual relationships that can be later promoted as domain knowledge and converted in OWL[1] format. The paper details each class of patterns, and presents the accuracy of each pattern prior to any filtering. Our results show that filtering techniques should be used in a separate sieve on top of the ODPs to improve the accuracy of the extraction.

2 Related Work

ODP are quite recent design patterns whose objective is to create modeling solutions to well-identified problems in ontology engineering and thus promote good design. In this paper, we are mostly interested in one subclass of ODPs: Lexico-syntactic patterns (LSPs). In particular LSPs are meant to identify which patterns in texts correspond to logical constructs of the OWL language. This type of patterns is essential from a semi-automatic ontology engineering point of view. In fact, as soon as the domain becomes a real-world problem, it starts to be difficult and very expensive to manually design an ontology.

In general LSPs are a widely used method in text mining and can be traced back to the work of [4] for hyponymy extraction. There have been several attempts to use LSPs for ontology learning [2, 3, 6, 7], relation extraction [7] or for axiom extraction [3]. Among the most similar works to ours are SPRAT [7] and [2] which propose various patterns for ontology learning and population. One of the peculiarities of our approach is that we designed purely syntactic patterns that examine a bigger number of linguistic constructs (e.g. relative clause modifiers, adjectival complements, copula, etc.) than what is available in the state of the art to extract information from text. Moreover, our patterns are based solely on a dependency grammar combined with parts-of-speech tagging. We used a similar approach in a previous work [8] on semantic analysis but the aim was not the extraction of triples for ontology learning and the patterns themselves were not structured and conceived in the same way. Finally, there are also various LSPs identified on the ODP portal[2] but to the best of our knowledge, the majority of the patterns in this paper are new with respect to the listed LSPs. Overall, 29 over the 31 patterns presented in this paper are not listed on the ODP portal. There is one common pattern for object property extraction (*nsubj-dobj*) which is already widely used in the information extraction field and one common pattern for hierarchical relations extraction (*nsubj-cop*). Finally, in one case, there is a similarity between one LSP used to extract sub-classes/super-classes relationships and one of our patterns for hierarchical relationship extraction (with the use of the expression *including* instead of *include*). In any case, one major difference is the use of dependency relations in OntoCmaps.

[1] http://www.w3.org/TR/owl-features/
[2] http://ontologydesignpatterns.org/wiki/Category:ProposedLexicoSyntacticOP

3 Lexico-Syntactic Patterns in OntoCmaps

3.1 OntoCmaps

OntoCmaps is an ontology learning tool that takes unstructured texts about a domain of interest as input [6]. OntoCmaps is essentially based on two main stages: a knowledge extraction step which relies on syntactic patterns to extract candidate triples from texts, and a knowledge filtering step which acts as a sieve to identify relevant triples among these candidates. Since this paper focuses on the knowledge extraction part, this section presents the formalisms and tools used during the extraction stage.

As aforementioned, OntoCmaps patterns are mainly syntactic patterns which use a dependency grammar formalism and part-of-speech tagging [9]. The dependency analysis is obtained through the Stanford Parser [5] which defines a grammatical relations hierarchy and outputs dependencies (we use the collapsed dependencies). A dependency parse represents a set of grammatical relations (from this hierarchy) that link each pair of related words in a sentence. Several examples of dependency parses are provided in the following sections.

3.2 Syntactic Patterns

Patterns define specific syntactic configurations that link variables, constrained by given parts-of-speech, using grammatical dependencies. Parts of speech constraints allow filling in the variables with the right types of grammatical categories and therefore are essential for the accuracy of the extraction [8]. Parts of speech are defined in the Penn Treebank II[3].

Some patterns might overlap and are organized into a pattern hierarchy to trigger the more detailed patterns first. When a parent pattern is instantiated, all its children are disqualified for the current sentence, to avoid the extraction of meaningless fragments. Patterns are interpreted (using Java methods) to extract triples that can represent candidate domain ontological relationships. Triples also let OntoCmaps identify potentially relevant domain terms (i.e. content words).

The transformation rules (which can be considered as Transformation ODPs) focus on triples to enable mappings with the OWL-DL language. OWL DL is much more limited than natural language. Consequently, various syntactic configurations do not have any equivalent in OWL-DL. For example, verbs tense cannot be represented. There are general conventions that are followed in OntoCmaps for generating possible mappings: 1) Nouns and combinations of nouns, adjectives and adverbial modifiers are converted into potential candidate classes; 2) Proper nouns are converted into named entities (potential instances); 3) Comparative adjectives potentially map to an OWL Object Property when domain and range are classes; 4) Transitive verbs map to potential OWL Object Properties when domain and range are classes. They can also map to data types properties if the range is not considered as a class; 5) Negation on a

[3] http://www.cis.upenn.edu/~treebank/

verb between two identified classes maps to the OWL complement construct; 7) Verb tenses, modals and particles are all aggregated in the label of a potential OWL object property (verb); 8) The noun following a possessive pronoun is translated into and OWL Object Property or a data type property; 9) Determiners, quantifiers, comparative and superlative adverbs are ignored in OntoCmaps at this point.
There is no predefined meaning assigned to any of the extracted terms and relationships. Terms and relationships labels might have various morphological forms but are all related to their root lemma. Therefore, various morphological structures with the same root all relate to the same relation or term. Semantics is left underspecified or more specifically specified by the domain context, since OntoCmaps takes a domain corpus as input. If there are triples related to the term "bank", then whether it is the financial institution or the side of a river will be determined by the input corpus and by the other extracted relationships. The following sections details the patterns used by OntoCmaps. A pattern is represented using the following convention:

Grammatical relation (Head-Index / POS, Dependent-Index/POS) → Transformation

- Grammatical relation represents a dependency relation;
- Head and Dependent are variable names;
- POS represents a part-of-speech. Note that we use the generic part-of-speech NN for all the noun parts-of-speech (NN, NNS) and the generic part-of-speech VB for all the verbal parts-of-speech (VB, VBD, VBG, VBN, VBP and VBZ);
- Index represents the position of the word in the sentence;
- Transformation describes the resulting expression when this pattern is instantiated.

3.3 Expression Extraction

Simple and multi-word expressions (MWE) are considered candidate domain terms if they occur in lexico-syntactic ODPs for hierarchical and conceptual relationship extraction. The first step in OntoCmaps consists of aggregating MWE to generate a new dependency graph composed of MWE linked by grammatical relations (Table 1). The most common MWE are obtained through the patterns (1), (2) and (3).

Table 1. Multi-word expressions patterns

Pattern	Example
`nn(X/NN, Y/NN) → Y_X (1)`	nn(Systems-3/NN, Computer-1/NN), nn(Systems-3/NN, Operating-2//NN) →Computer operating systems
`amod(X/NN, Y/JJ)→Y_Z (2)`	amod(intelligence-2/NN, Artificial-1/JJ) → Artificial Intelligence
`nn(X,Y), amod(X,Z) → Z_Y_X (3)`	amod(systems-3/NN, Intelligent-1/JJ), nn(systems-3/NN, computing-2/NN)→Intelligent computing systems Note that Pattern (3) is a combination of (1) and (2).

`advmod(X/VBN, Y/RB), dobj(X/VBN, Z/NN)` → `Y_X_Z (4)`	advmod(modified-2/VB, Experimentally-1/RB), dobj(modified-2/VB, cell-3/NN)→ Experimentally modified cell
`prep_of/IN (X/NNP, Y/NNP)→ X_of_Y(5)`	prep_of/IN (University-2//NNP, Toronto-4/ NNP)→ University of Toronto
`prep_of /IN(X/NN, Y/NN)→ X_of_Y (6)`	prep_of/IN(page-2/NN, book-5/NN)→ Page of book Note that another possible transformation could be Y X (e.g. book page).

We also designed two patterns for multi-word expressions containing the preposition *of* (5) (6). Pattern (5) extracts a named entity and is useful for ontology population (rather than learning). The type of multi-word expressions in (6) can be tricky, as the Y part can represent the domain concept and the X part might be only an attribute (e.g. the color of the car) or a part (the wheel of the car). However, if this MWE is not important for the domain, it is likely that it will be sieved during the filtering stage.

3.4 Relationship Extraction

Relationship extraction in OntoCmaps refers to hierarchical relationships extraction (aka taxonomy or hyponymy) and conceptual relationships extraction (OWL Object Properties). Relationship extraction is run after few other operations, mainly the aggregation of multi-words expressions, the distributive interpretation of conjunctions and the removal of function words such as determiners and quantifiers. These prior operations produce a modified dependency graph used as input for relationships extraction.

Hierarchical Relationship Extraction.

There have been many works [2, 4, 10] in hierarchical relationships extraction using patterns. In OntoCmaps, hierarchical relationships are mapped to subclasses in OWL-DL.

OntoCmaps reuses some of Hearst's patterns (patterns (7) (8) in Table 2) using the dependency grammar formalism, parts-of speech, and transformation rules. We also create a hierarchical relationship from the multi-word expression pattern (3) (in Table 1) which involves a noun compound modifier and an adjectival modifier (pattern 9, Table 2) and from the multi-word expression pattern 4 (Table 1) thus obtaining pattern (10) (Table 2). Finally, we designed one pattern based on the copula (Pattern (11), Table 2).

Table 2. Hierarchical Relations Patterns

Pattern	Example
`prep_such_as/IN(X/NNS,`	Animals such as monkeys and apes prep_such_as/IN(animals-1/NNS, monkeys-

`Y/NNS)→is-a(Y, X) (7)`	4/NNS), prep_such_as/IN(animals-1/NNS, apes-6/NNS) → *is_a (monkeys, animals); is-a(apes, animals)*
`prep_including/IN(X/NNS, Y/NNS)→is-a(Y, X) (8)`	All buildings, including houses and castles… prep_including/IN(buildings-2/NNS, houses-5/NNS*)→ is-a(houses, buildings); is-a(castles, buildings)*
`nn(X,Y), amod(Y,Z)→ is-a (Y, X) (9)`	Intelligent computing systems… amod(systems-3, Intelligent-1), nn(systems-3, computing-2*)→ is-a (Intelligent computing systems, computing systems)→ is-a(Computing systems, systems)*
`advmod(X/VBN, Y/RB), dobj(X/VBN, Z/NN) → is-a(Y_X_Z, Z)(10)`	Genetically modified food advmod(modified-2, Genetically-1), dobj(modified-2, food-3)→ *is-a(Genetically modified food, food)*
`nsubj(X/NNS, Y/NNS), cop(X/NNS, Z/VBP)→ is-a(Y,X) (11)`	Pinguins are birds nsubj(birds-3/NNS, Pinguins-1/NNS), cop(birds-3/NNS, are-2/VBP*)→ is-a(Pinguins, Birds)*

Conceptual Relationships Extraction.

Conceptual relationships refer to OWL Object Properties with a domain and range. These relationships are among the most difficult to extract. We propose dependency-based patterns that are divided into four main categories: main clauses (containing a nominal subject nsubj), passive clauses (containing a passive nominal subject), relative clauses (containing a relative clause modifier) and finally other clauses which group certain constructs not belonging to the other categories.

Main clauses.

Main clauses are organized around the main verb of the sentence. Pattern (12) has been already referenced in the ODP portal. Pattern (13) enriches Pattern (12) with a preposition attached to the main verb and allows the creation of two triples. Patterns (16-18) use the xcomp relationship (which indicates a clausal complement with an external subject) to create a relationship between the main subject of the sentence (nsubj) and its direct object (Pattern (16) and (18)) or its related agent (Pattern (17)). Finally, Pattern (14) and (15) create a relationship between the nominal subject and the object of the preposition.

Table 3. Main clause patterns for conceptual relationship extraction

Pattern	Example
`nsubj(X/VB, Y/NN),`	Content packaging can define content organizations.

dobj(X/VB, Z/NN)→X(Y,Z) (12)	→ *can define(content packaging, content organizations)*
nsubj(X/VB, Y/NN), dobj(X/VB, Z/NN), prep_K(X/VB, A/NN)→ X_Z_K(Y, A), X(Y,Z)(13)	AICC has submitted CMI001 to the IEEE. → *has submitted(aicc , cmi001)* *has submitted cmi001 to (aicc ,ieee)*
nsubj(X/JJ, Y/NN), prep_K(X/JJ, A/NN)→ X_K(Y,A) (14)	The RTE describes the LMS requirements for managing the runtime environment such as standardized data model elements used for passing information relevant to the learner's experience with the content). → *relevant_to(information, experience)*
Nsubj(X/JJ, Y/NN), cop(X/JJ, V/VB) prep_P(X/JJ, Z/NN)→X_P(Y,Z)(15)	These branches are visible to the LMS. → *visible to(branch, lms)*
nsubj(X/JJ, Y/NN), cop(X/JJ, C/VB), xcomp(X/JJ, V/VB), dobj(V/VB, Z/NN), → X_V(Y, Z) (16)	The Sequencing Control Choice element indicates that the learner is free to choose any activity in a cluster in any order without restriction. → *free to choose (learner, activity)*
nsubj(X/JJ, Y/NN) xcomp(X/JJ, V/VB) agent(V/VB, Z/NN)→ X_V(Y,Z) (17)	The difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples. → *too large to be covered by (set of possible behavior, set of observed examples)*
Nsubj(X/VB, Y/NN), dobj(X/VB, V/NN) xcomp(X/VB, Z/VB), dobj(Z/VB, N/NN)→ *X_V_Z(Y, N)*(18)	SCORM recognizes that some training resources may contain internal logic to accomplish a particular learning task → *may contain internal logic to accomplish (training resource, learning task)*

Passive clauses.

Passive clauses allow the extraction of conceptual relationships (Table 4) and sometimes their inverse property. For instance, pattern (19) (Table 4) can be used to define such an inverse property for the relation *defined set of information - can be tracked by - lms environment* by creating an OWL inverseOf relation: *lms environment - can track - defined set of information.*

Table 4. Passive clauses patterns for conceptual relationship extraction

Pattern	Example
`Nsubjpass(V/VB, Y/N), Agent(V/VB, Z/N)→ V(Y,Z) (19)`	The purpose of establishing a common data model is to ensure that a set of information can be tracked by different LMS environments. → *can be tracked by (defined set of information, lms environment)*
`Nsubjpass(V/JJ, X/NN), cop(V/JJ, V/VB), prep_P(V/JJ, Y/NN)→ V_P(X,Y)(20)`	Sequencing information is defined on the Activities and is external to the training resources associated with those Activities. → *external to (sequencing information, training resource)*
`Nsubjpass(V/VB, X/NN), prep_P(V/VB, Y/NN)→ V_P(X,Y) (21)`	Metadata can be applied to Assets. → *can be applied to (metadata, assets)*
`Nsubjpass(V/VB,X/NN), dobj(V/VB,Y/NN)→ V(X,Y)(22)`	The data model element names shall be considered reserved tokens. → *shall be considered(data model element names, reserved tokens)*

Relative Clauses.

Relative clauses (see Table 5) are generally neglected by similar pattern-based approaches, as they are often distant from their main subject. The relationships created by our patterns in this category have generally lengthy labels, but they allow us to find links between two candidate concepts that might be otherwise neglected.

Table 5. Relative clauses patterns for conceptual relationship extraction

Pattern	Example
`rcmod(X/NN, Y/VB), dobj(Y/VB, Z/NN), prep_P(Y/VB, Q/NN) → Y_Z_P(X, Q), Y(X, Z) (23)`	Learning Management System is a software that automates event administration through a set of services. → *automates_event_administration_through (Software, set_of_services)*
`Nsubj(X/NN, Y/NN), rcmod(X/NN, V/VB), dobj(V/VB, Z/NN)→V(Z,Y)(24)`	→ *automates(learning management system, event administration)*
`Nsubj(X/NN, Y/NN), rcmod(X/NN, V/VB), dobj(V/VB, Z/NN), Prep_P(V/VB, Q/NN)→V_Z_P(Y, Q)(25)`	→ *automates event administration through (Learning management system, set of services)*

Rcmod(X/NN, V/VB), xcomp(V1/VB,V2/VB), dobj(V2/VB, Y/NN), prep_P(V2/VB, Z/NN)→ V1_V2_Y_P (X, Z) (26)	"1484.11.1" is a standard that defines a set of data model elements that can be used to communicate information from a content object to an LMS. → *can be used to communicate information from (set of data model element, content object)*
rcmod(X/NN, V/VB), dobj(V/VB, Z/NN)→ V(X,Z) (27)	This keyword data model element can only be applied to a data model element that has children. → *has (data model element, children)*

Relative clauses patterns are focused around the dependency relationship *rcmod* and enable making links between a main subject and the *rcmod* direct object (Pattern 24) or a preposition (Pattern (23) and (25)) in the relative clause. Pattern (26) makes a link between the noun in a relative clause modifier with the clausal complement xcomp, and finally Pattern (27) links the noun of the relative clause modifier to its direct object.

Other clauses.

Finally, there are some clauses around infinitival modifiers (*infmod*) and participial modifiers (*partmod*) that modify their noun phrase and that allow us to create conceptual relationships (Table 6) when they have a direct object or a preposition. Note again that pattern 31 (Table 6), which has an agent dependency, can lead to an OWL inverse property similar to the one explained in the passive clauses.

Table 6. Other clauses patterns for conceptual relationship extraction

Infmod(X/NN,Y/VB), Dobj(Y/VB,Z/NN)→Y(X,Z) (28)	This value can be requested by the SCO to determine the next index position.→ *determine (sco, next index position)*
Partmod(X/NN,V/VB),dobj(V/VB, Y/NN)→ V(X,Y) (29)	A SCO can communicate with an LMS using the SCORM RTE → *using(lms, scorm rte)*
Partmod(X/NN, V/VB), prep_P(V/VB, Y/NN) → V_P(X,Y) (30)	...to describe the components used in a learning experience. → *used in (components, learning experience)*
Partmod(X/NN, V/VB), agent(V/VB, Y/NN)→V(X,Y) (31)	All data model elements described by SCORM are …. *described by (data model elements, scorm)*

4 Evaluation

In order to evaluate our patterns, we essentially relied on two corpora used in previous experiments [6]: the SCORM corpus, which is a set of manuals on the SCORM eLearning standard and the AI corpus, which is a set of Wikipedia pages about artificial intelligence. We previously generated and validated two OWL ontologies from these corpora and we consider them as our gold standards (GSs). Details about these GSs can be found in [6][4]. We then calculated the precision of the various patterns based on these GSs.

Precision= number of generated relationships or concepts per pattern (A) / number of relationships or concepts in (A) that exist in the GS

Tables 7-11 report the various results of each pattern category[5]. Each table presents, for each pattern in the category, the precision of the extracted relationships and concepts in both corpora. One must note that some of the relationships extracted by patterns were perfectly valid (from a lexical point of view) but were not found in the GS, thus reducing the reported precision. Another point is that concepts are simple or multi-words expressions that occur in a relationship. Therefore, we were able to calculate concept precision as well by identifying how many concepts involved in the extracted relationships were in the GSs.

Table 7. Precision of Hierarchical Relationships Patterns.

Pattern	Relations SCORM	Relations AI	Concepts SCORM	Concepts AI
Pattern (7)	47.46	93.75	79.45	84.37
Pattern (11)	74.63	56.76	91.29	72.79
Pattern (8)	100.00	76.92	100.00	80.77
Average	**74.03**	**75.81**	**90.25**	**79.31**

We can notice that the precision of hierarchical relationships and their corresponding concepts is quite high (Table 7).

Table 8. Precision of Conceptual Relationships Patterns (Main Clauses)

Pattern	Relations SCORM	Relations AI	Concepts SCORM	Concepts AI
Pattern (12)	52.16	24.10	81.75	57.45
Pattern (16)	75.00	0	100.00	33.33
Pattern (15)	50.75	35.29	79.58	61.67
Pattern (13)	50.00	26.25	78.24	49.51
Pattern (14)	36.31	28.33	79.49	62.62

[4] Available at http://azouaq.athabascau.ca/goldstandards.htm

[5] Extraction examples for each pattern can be found at http://azouaq.athabascau.ca/experiments/wop2012/SCORMPatterns_WOP2012.xls and AI-Patterns_WOP2012.xls

Pattern (17)	0	0	0	0
Pattern (18)	0	0	80.43	90
Average	**37.74**	**16.28**	**71.35**	**50.65**

Interestingly, passive clauses (Table 9) and other clauses (Table 11) achieve a better performance for relationships precision than main clauses (Table 8) which get approximately the same results as relative clauses (Table 10).

Table 9. Precision of Conceptual Relationships Patterns (Passive Clauses).

Pattern	Relations SCORM	Relations AI	Concepts SCORM	Concepts AI
Pattern (19)	69.56	25.00	74.00	56.25
Pattern (22)	60.00	66.67	88.00	100.00
Pattern (21)	58.45	44.78	84.16	66.67
Pattern (20)	75.00	*na*	95.00	*Na*
Average	**65.75**	**45.48**	**85.29**	**74.31**

Table 10. Precision of Conceptual Relationships Patterns (Relative Clauses).

Pattern	Relations SCORM	Relations AI	Concepts SCORM	Concepts AI
Pattern (23)	41.18	56.25	62.07	66.67
Pattern (25)	0	0	90.48	60.00
Pattern (24)	73.33	50.00	93.90	76.47
Pattern (26)	0	0	85.71	0
Pattern (27)	50.62	21.62	86.39	56.52
Average	**33.02**	**25.57**	**83.71**	**51.93**

Table 11. Precision of Conceptual Relationships Patterns (Other Clauses).

Pattern	Relations SCORM	Relations AI	Concepts SCORM	Concepts AI
Pattern (29)	43.18	25.00	80.58	70.00
Pattern (30)	60.58	54.54	86.04	73.58
Pattern (28)	38.77	37.50	82.10	80.77
Average	**47.51**	**26.85**	**82.91**	**74.78**

These results give a general idea on the Precision of the lexico-syntactic patterns. As we have previously mentioned, and as the results confirm it, there is a need to filter the various extractions using statistical and/or graph-based metrics [6]. The most frequent patterns were *nsubj-dobj*, *nsubjpas-prep*, *nsubj-dobj-prep*, *nsubj-prep* and *partmod-prep*. The most precise (but scarcer) patterns, without any filtering, were hierarchical patterns. One important observation is the quite high precision of concepts even without filtering. Regarding relationships, it is possible to imagine that if concepts of interest are known upfront, then these patterns will be very useful for discovering relationships between these predefined concepts. This will be tackled in

future work. We also created few patterns for attributes extraction involving possessives or nominal subject and copula with adjectives. However, the way to translate these attribute relationships into OWL-DL was not straightforward.

5 Conclusion

This paper presented a list of the main patterns used in OntoCmaps, our ontology learning tool. These patterns target specific syntactic structures in a dependency representation and are useful for the extraction of multi-word expressions and triples that can be later translated into OWL classes and properties. There were some simplifying assumptions made in OntoCmaps, mainly the removal of determiners and the lack of co-reference resolution that should be included in future work. In this current state, our patterns represent a good starting base that any researcher in text mining might use and especially the ontology learning community which lacks clear and reusable design patterns. Overall, future efforts will tackle how and if a more fine-grained semantic analysis would be beneficial to the ontology learning task. Another future task will be to extend the coverage of our patterns by extracting frequently occurring syntactic structures using machine learning methods. Finally, one of the lessons learned in this paper is that such pattern-based extraction should necessarily be coupled with a filtering mechanism to increase the precision of the extractions.

Acknowledgments. This research was funded by the NSERC Discovery Grant Program.

References

1. Blomqvist. E.: Semi-automatic Ontology Construction based on Patterns. PhD Thesis. Linköping University, Department of Computer and Information Science (2009)
2. Presutti, V. et al.: A Library of Ontology Design Patterns: Reusable Solutions for Collaborative Design of Networked Ontologies. NeOn D2.5.1 (2008)
3. Völker, J., Hitzler, P. & Cimiano, P.: Acquisition of OWL DL axioms from lexical resources. In: Proc. of the 4th European Semantic Web Conf., pp.670-685, Springer (2007)
4. Hearst. M.: Automatic Acquisition of Hyponyms from Large Text Corpora. In Proc. of the 14th International Conf. on Computational Linguistics, pp.539-545 (1992).
5. De Marneffe, M-C , MacCartney B. & Manning, C. D.: Generating Typed Dependency Parses from Phrase Structure Parses. In Proc. of *LREC,* pp. 449–454 (2006).
6. Zouaq, A., Gasevic, D. & Hatala, M.: Towards open ontology learning and filtering. Inf. Syst. 36(7): 1064-1081 (2011)
7. Maynard, D.; Funk, A. & Peters, W.: Using Lexico-Syntactic Ontology Design Patterns for ontology creation and population, CEUR-WS.org, (2009).
8. Zouaq, A., Gagnon, M. & Ozell, B.: Semantic Analysis using Dependency-based Grammars and Upper-Level Ontologies, *International Journal of Computational Linguistics and Applications*, 1(1-2): 85-101, Bahri Publications (2010).
9. Toutanova, K., Klein, D., Manning, D. & Singer. M.: Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proc. of HLT-NAACL*, pp. 252-259 (2003).
10. Snow, R., Jurafsky, D. and Ng, A. Y.: *Learning syntactic patterns for automatic hypernym discovery.* In: Advances in Neural Information Processing Systems, pp.1297-1304 (2004).