Automatically Harvesting and Ontologizing Semantic Relations

Patrick PANTEL [a] and Marco PENNACCHIOTTI [b]

[a] *Information Sciences Institute,University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90292 pantel@isi.edu*

[b] *Dept. of Computational Linguistics, Saarland University, Germany pennacchiotti@coli.uni-sb.de*

Abstract. With the advent of the Web and the explosion of available textual data, it is key for modern natural language processing systems to access, represent and reason over large amounts of knowledge in semantic repositories. Separately, the knowledge representation and natural language processing communities have been developing representations/engines for reasoning over knowledge and algorithms for automatically harvesting knowledge from textual data, respectively. There is a pressing need for collaboration between the two communities to provide large-scale robust reasoning capabilities for knowledge rich applications like question answering. In this chapter, we propose one small step by presenting algorithms for harvesting semantic relations from text and then automatically linking the knowledge into existing semantic repositories. Experimental results show better than state of the art performance on both relation harvesting and ontologizing tasks.

Keywords. knowledge acquisition, relation extraction, ontology learning

1. Introduction

With the advent of the Web and the explosion of available textual data, it is key for modern Natural Language Processing (NLP) systems to access, represent and reason over large amounts of knowledge contained in semantic repositories.

Separately, the knowledge representation (KR) community has developed many formal ontologies for use in various reasoning tasks such as planning and theorem proving, and the natural language processing (NLP) community has developed several algorithms for automatically harvesting knowledge from textual resources. Most mined resources from NLP consist of very large but noisy and unstructured knowledge, making their use in KR reasoning engines futile. Knowledge rich applications such as question answering and information extraction would benefit greatly from the reasoning power of the KR community and the breadth of knowledge extracted from the NLP community. There is therefore a pressing need for the NLP community to not only harvest knowledge from text, but also to link this knowledge into semantic repositories over which KR reasoning engines can execute.

In this chapter, we present algorithms for both extracting semantic relations from textual resources and for linking, or *ontologizing*, them into a semantic repository.

1.1. Exploiting Knowledge Resources

Recent attention to knowledge-rich problems such as question answering [1] and textual entailment [2] has encouraged natural language processing researchers to develop algorithms for automatically harvesting semantic resources. With seemingly endless amounts of textual data at our disposal, we have a tremendous opportunity to automatically grow semantic term banks and ontological resources.

Knowledge resources can be mainly divided in two types: *textual resources* and *structured resources*. *Textual resources* include linguistic text collections, ranging from large generic repositories such as the Web to specific domain texts such as collections of texts or books on specific subjects. These repositories contain a large and ever growing amount of information expressed *implicitly* in natural language texts. These resources greatly vary in size, from the terabytes of data on the Web to the kilobytes of textual material in electronic books. Structured resources consist of repositories in which knowledge is *explicit* and organized in lists or graphs of entities. In contrast with textual resources, structured resources are used to explicitly represent domain and generic knowledge, making their inherent knowledge directly usable in applications. *Structured resources* vary largely on their degree of internal structuring, and can be accordingly divided in two different classes: *semantic repositories* and *lexical resources*. The first class is formed by highly structured resources that usually organize knowledge at a *conceptual* level (e.g., concepts, relations among concepts, situation types) or at a *sense* level (word senses and relations among senses). Ontologies such as Mikrokosmos [3,4], DOLCE [5] and SUMO [6], and situation repositories such as FrameNet [7] are good examples of the former, while WordNet [8] is an example of the latter. *Lexical resources* are less structured resources such as thesauri, lists of facts, lexical relation instances, lists of paraphrases, and other flat lists of lexical objects. These resources usually organize knowledge at a pure *lexical* level, and are in most cases built by using automatic or semi-automatic techniques.

Two main issues must be addressed in order to use knowledge resources in applications: extract the implicit knowledge in textual resources (*knowledge harvesting*), and make the knowledge of both textual and structured resource usable (*knowledge exploitation*).

Regarding *knowledge harvesting*, harvesting algorithms are used to analyze textual repositories and extract knowledge in the form of lexical resources. NLP researchers have developed many algorithms for mining knowledge from text and the Web, including facts [9], semantic lexicons [10], concept lists [11], and word similarity lists [12]. Many recent efforts have also focused on extracting binary semantic relations between entities, such as entailments [13], *is-a* [14], *part-of* [15], and other relations. Relational knowledge is in fact crucial in many applications. Unfortunately, most relation extraction algorithms suffer from many limitations. First, they require a high degree of supervision. Secondly, they are usually limited in breadth (they cannot be easily applied to different corpus sizes and domains) and generality (they can harvest only specific types of relations).

So far, little attention has been spent on the issue of *knowledge exploitation*. As Bos [16] outlined, whilst lexical resources are potentially useful, their successful use in applications has been very limited due to a variety of problems. For example, question answering (QA) systems based on logical proving could in theory improve their performance by simply exploring knowledge in lexical resources which have been acquired indipendently and semantic repositories. For instance, suppose a QA system must answer the following question:

"When did James Dean die?"

Suppose the system could rely on a lexical resource formed by a list of entailment rules. The lexical resource could contain the entailment $kill(X,Y) \rightarrow die(Y)$. The system could then answer *"1955"*, by examining the Web and finding the snippet *"In 1955, actor James Dean was killed in a two-car collision near Cholame, Calif"*.

Consider the following question:

"Who was Horus' father?"

A system could answer *"Osiris"* from the snippet *"It also hosted statues of Amon's wife, Mut, the goddess Isis, her husband, Osiris, and their son Horus"*, by using a generic world knowledge ontology containing the fact:

$$\forall x(husband(x) \rightarrow male(x))$$
$$\forall x \forall y(son(x) \wedge of(x,y) \wedge male(y) \rightarrow father(y) \wedge of(y,x))$$

The main reasons that limit the exploitation of existing resources stem from the nature of semantic repositories and lexical resources. Although rich in structure and precision, semantic repositories are difficult to use since they are built by hand and are therefore limited in size and scope. In contrast, lexical resources represent a very large amount of knowledge, but they suffer from low precision and structure.

1.2. Harvesting and Ontologizing Knowledge Desiderata

In order to leverage knowledge resources in NLP applications, it is necessary to improve knowledge harvesting algorithms and to integrate the different types of resources in a coherent framework (e.g., a semantic repository such as an ontology or term bank). An ideal framework for knowledge harvesting and exploitation should then guarantee the following desired properties:

- *Generality*. Knowledge harvesting should be able to extract as many relation types as possible.
- *Minimal supervision*. Knowledge harvesting should be carried out using little or no human intervention.
- *Breadth*. Harvesting algorithms should be adaptable to different corpus sizes, in order to successfully extract knowledge from both large textual resources such as the Web and small ones.
- *Precision*. Harvested knowledge must be precise. As lexical resources are usually very noisy, they can be made more precise by both improving the harvesting algorithms and by filtering erroneous information during the linking process to a semantic repository.
- *Domain knowledge coverage*. Harvested knowledge must cover all the domain knowledge.
- *Closeness to language*. As applications work on linguistic expressions, it is crucial to map conceptual/sense knowledge to language. This can be achieved by linking concepts/senses and relations in a semantic repository to terms and term relations in a lexical resource.

- *Structure*. The structure of a semantic repository is a key aspect to expand the lexical knowledge embedded in lexical resources and make them usable in applications. For example a simple list of *part-of* relation instances can be expanded by using generalizations or synonymy information enclosed in a semantic repository such a WordNet.

1.3. Harvesting and Ontologizing Knowledge in Practice

In this chapter, we present a pipeline of two systems which form a complete and coherent framework for knowledge harvesting and exploitation, by addressing the above mentioned desired properties. In particular, our systems aim at addressing the issue of extracting and ontologizing *relational knowledge*, which is an important component of many NLP applications, as outlined in Section 1.1. The first of these two systems is called *Espresso* and is described in Section 3. *Espresso* is a general-purpose, broad, and accurate corpus harvesting algorithm requiring minimal supervision. The main algorithmic contribution is a novel method for exploiting *generic patterns*, which are broad coverage noisy extraction patterns - i.e., patterns with high recall and low precision. Insofar, difficulties in using these patterns have been a major impediment for minimally supervised algorithms resulting in either very low precision or very low recall. We propose a method to automatically detect generic patterns and to separate their correct and incorrect instances. The key intuition behind the algorithm is that given a set of *reliable* (high precision) patterns on a corpus, correct instances of a generic pattern will fire more with reliable patterns on a very large corpus, like the Web, than incorrect ones. Previous work like Girju et Al. [15] that has made use of generic patterns through filtering has shown both high precision and high recall, at the expensive cost of much manual semantic annotation. Minimally supervised algorithms, like [17,18], typically ignore generic patterns since system precision dramatically decreases from the introduced noise and bootstrapping quickly spins out of control.

Secondly, is Section 4, we propose a system which adopts two alternative algorithms for ontologizing binary semantic relations into WordNet. Formally, given an instance (x, r, y) of a binary relation r between terms x and y, the ontologizing task is to identify the WordNet senses of x and y where r holds. For example, the instance *(proton, PART-OF, element)* ontologizes into WordNet as *(proton#1, PART-OF, element#2)*. The first algorithm that we explore, called the *anchoring approach*, was suggested as a promising avenue of future work by Pantel [19]. This bottom up algorithm is based on the intuition that x can be disambiguated by retrieving the set of terms that occur in the same relation r with y and then finding the senses of x that are most similar to this set. The assumption is that terms occurring in the same relation will tend to have similar meaning. We here propose a measure of similarity to capture this intuition. In contrast to anchoring, our second algorithm, called the *clustering approach*, takes a top-down view. Given a relation r, suppose that we are given every conceptual instance of r, i.e., instances of r in the upper ontology like *(particles#1, PART-OF, substances#1)*. An instance (x, r, y) can then be ontologized easily by finding the senses of x and y that are subsumed by ancestors linked by a conceptual instance of r. For example, the instance *(proton, PART-OF, element)* ontologizes to *(proton#1, PART-OF, element#2)* since $proton\#1$ is subsumed by *particles* and *element#2* is subsumed by *substances*. The problem then is to automatically infer the set of conceptual instances. For this purpose, we develop a clustering algorithm for generalizing a set of relation instances to conceptual instances by looking up the WordNet hypernymy hierarchy for common ancestors, as specific as possible, that subsume as many instances as possible. An instance is then attached to those senses that are subsumed by the highest scoring conceptual instances.

In Section 5 we report a complete experimental analysis of both systems. Experimental evidence demonstrates that our two systems are successful in harvesting and ontologizing relational knowledge by outperforming similar state of the art approaches.

2. Relevant Work

In this section, we review previous work in both relational knowledge harvesting and ontologizing.

2.1. Relational Knowledge Harvesting

To date, most research on relation harvesting has focused on *is-a* and *part-of*. Approaches fall into two categories: pattern- and clustering-based.

Most common are *pattern-based approaches*. Hearst [17] pioneered using patterns to extract hyponym (*is-a*) relations. Manually building three lexico-syntactic patterns, Hearst sketched a bootstrapping algorithm to learn more patterns from instances, which has served as the model for most subsequent pattern-based algorithms.

Berland and Charniank [20] proposed a system for *part-of* relation extraction, based on the Hearst [17] approach. Seed instances are used to infer linguistic patterns that are used to extract new instances. While this study introduces statistical measures to evaluate instance quality, it remains vulnerable to data sparseness and has the limitation of considering only one-word terms.

Improving upon Berland and Charniank [20], Girju et Al. [15] employ machine learning algorithms and WordNet [8] to disambiguate *part-of* generic patterns like *"X's Y"* and *"X of Y"*. This study is the first extensive attempt to make use of generic patterns. In order to discard incorrect instances, they learn WordNet-based selectional restrictions, like *"X(scene#4)'s Y(movie#1)"*. While making huge grounds on improving precision/recall, heavy supervision is required through manual semantic annotations.

Ravichandran and Hovy [14] focus on scaling relation extraction to the Web. A simple and effective algorithm is proposed to infer surface patterns from a small set of instance seeds by extracting substrings relating seeds in corpus sentences. The approach gives good results on specific relations such as *birthdates*, however it has low precision on generic ones like *is-a* and *part-of*. Pantel and et Al. [21] proposed a similar, highly scalable approach, based on an edit-distance technique, to learn lexico-syntactic patterns, showing both good performance and computational efficiency. *Espresso* uses a similar approach to infer patterns, but we make use of generic patterns and apply refining techniques to deal with a wide variety of relations.

Other pattern-based algorithms have been proposed by Riloff and Shepherd [10], who used a semi-automatic method for discovering similar words using a few seed examples, in KnowItAll [9] that performs large-scale extraction of facts from the Web, by Mann [22] who used part of speech patterns to extract a subset of *is-a* relations involving proper nouns, by Downey et Al. [23] who formalized the problem of relation extraction in a coherent and effective combinatorial model that is shown to outperform previous probabilistic frameworks, by Snow et Al. [24], and in co-occurrence approaches such as in Roark and Charniak [25]. Ciaramita et al.'s chapter in this book presents a very nice approach to learning structured arbitrary binary semantic relations which is fully unsupervised, domain independent and quite efficient since it ultimately relies on named-entity tagging and dependency parsing which can be both solved in linear time.

Clustering approaches have so far been applied only to *is-a* extraction. These methods use clustering algorithms to group words according to their meanings in text, label the clusters using its members' lexical or syntactic dependencies, and then extract an *is-a* relation between each cluster member and the cluster label. Caraballo [26] proposed the first attempt which used conjunction and apposition features to build noun clusters. Recently, Pantel and Ravichandran [18] extended this approach by making use of all syntactic dependency features for each noun. The advantage of clustering approaches is that they permit algorithms to identify *is-a* relations that do not explicitly appear in text, however they generally fail to produce coherent clusters from fewer than 100 million words; hence they are unreliable for small corpora.

2.2. *Ontologizing Knowledge*

Several researchers have worked on ontologizing semantic resources. Most recently, Pantel [19] defined the task of ontologizing a lexical semantic resource as linking its terms to the concepts in a WordNet-like hierarchy. He developed a method to propagate lexical co-occurrence vectors to WordNet synsets, forming ontological co-occurrence vectors. Adopting an extension of the distributional hypothesis [27], the co-occurrence vectors are used to compute the similarity between synset/synset and between lexical term/synset. An unknown term is then attached to the WordNet synset whose co-occurrence vector is most similar to the term's co-occurrence vector. Though the author suggests a method for attaching more complex lexical structures like binary semantic relations, he focuses only on attaching terms.

Basili et Al. [28] proposed an unsupervised method to infer semantic classes (WordNet synsets) for terms in domain-specific verb relations. These relations, such as *(x, EXPAND, y)* are first automatically learnt from a corpus. The semantic classes of x and y are then inferred using *conceptual density* [29], a WordNet-based measure applied to all instantiations of x and y in the corpus. Semantic classes represent possible common generalizations of the verb arguments. At the end of the process, a set of syntactic-semantic patterns are available for each verb, such as:

(social_group#1, expand, act#2)
(instrumentality#2, expand, act#2)

The method is successful on specific relations with few instances (such as domain verb relations) while its value on generic and frequent relations, such as *part-of*, was untested.

Girju et Al. [15] presented a highly supervised machine learning algorithm to infer semantic constraints on *part-of* relations, such as *(object#1, PART-OF, social_event#1)*. These constraints are then used as selectional restrictions in harvesting *part-of* instances from ambiguous lexical patterns, like *"X of Y"*. The approach shows high performance in terms of precision and recall, but, as the authors acknowledge, it requires large human effort during the training phase.

Others have also made significant additions to WordNet. For example, in eXtended WordNet [30], the glosses in WordNet are enriched by disambiguating the nouns, verbs, adverbs, and adjectives with synsets. Another work has enriched WordNet synsets with topically related words extracted from the Web [31]. Finally, the general task of word sense disambiguation [32] is relevant since there the task is to ontologize each term in a passage into a WordNet-like sense inventory. If we had a large collection of sense-tagged text, then our mining algorithms could directly discover WordNet attachment points at harvest time. However, since there is little high precision sense-tagged corpora, methods are required to ontologize semantic resources without fully disambiguating text.

3. Knowledge Harvesting: The Espresso Algorithm

Espresso is based on the framework adopted by Hearst [17]. It is a minimally supervised bootstrapping algorithm that takes as input a few seed instances of a particular relation and iteratively learns surface patterns to extract more instances. The key to *Espresso* lies in its use of *generic patterns*, i.e., those broad coverage noisy patterns that extract both many correct and incorrect relation instances. For example, for *part-of* relations, the pattern *"X of Y"* extracts many correct relation instances like *"wheel of the car"* but also many incorrect ones like *"house of representatives"*.

The key assumption behind *Espresso* is that in very large corpora, like the Web, correct instances generated by a generic pattern will be instantiated by some *reliable patterns*, where reliable patterns are patterns that have high precision but often very low recall (e.g., *"X consists of Y"* for *part-of* relations). In this section, we describe the overall architecture of *Espresso*, propose a principled measure of reliability, and give an algorithm for exploiting generic patterns.

3.1. System Architecture

Espresso iterates between the following three phases: *pattern induction*, *pattern ranking/selection*, and *instance extraction*. The algorithm begins with seed instances of a particular binary relation (e.g., *is-a*) and then iterates through the phases until it extracts τ_1 patterns or the average pattern score decreases by more than τ_2 from the previous iteration. In our experiments, we set $\tau_1 = 5$ and $\tau_2 = 50\%$.

For our tokenization, in order to harvest multi-word terms as relation instances, we adopt a slightly modified version of the term definition given by Justeson and Katz [33], as it is one of the most commonly used in the NLP literature:

((Adj|Noun)+|((Adj|Noun)(NounPrep)?)(Adj|Noun)*)Noun*

This term defintion allows to capture both simple expressions like *underground economy*, and more complex ones like *Iraqi National Joint Action Committee for Reforms*.

3.1.1. Pattern Induction

In the *pattern induction* phase, *Espresso* infers a set of surface patterns P that connects as many of the seed instances as possible in a given corpus. Any pattern learning algorithm would do. We chose the state of the art algorithm described by Ravichandran and

Hovy [14] with the following slight modification. For each input instance $\{x, y\}$, we first retrieve all sentences containing the two terms x and y. The sentences are then generalized into a set of new sentences $S_{x,y}$ by replacing all terminological expressions by a terminological label, *TR*. For example:

"Because/IN HF/NNP is/VBZ a/DT weak/JJ acid/NN and/CC x is/VBZ a/DT y"

is generalized as:

"Because/IN TR is/VBZ a/DT TR and/CC x is/VBZ a/DT y"

Term generalization is useful for small corpora to reduce data sparseness. Generalized patterns are naturally less precise, but this is ameliorated by our filtering step described in Section 3.2.

As in the original algorithm, all substrings linking terms x and y are then extracted from $S_{x,y}$, and overall frequencies are computed to form P.

3.1.2. Pattern Ranking/Selection

In Ravichandran and Hovy [14], a frequency threshold on the patterns in P is set to select the final patterns. However, low frequency patterns may in fact be very good. In this work, instead of frequency, we propose a novel measure of pattern reliability, r_π, which is described in detail in Section 3.1.4. *Espresso* ranks all patterns in P according to reliability r_π and discards all but the top-*k*, where k is set to the number of patterns from the previous iteration plus one. In general, we expect that the set of patterns is formed by those of the previous iteration plus a new one. Yet, new statistical evidence can lead the algorithm to discard a pattern that was previously discovered.

3.1.3. Instance Extraction

In this phase, *Espresso* retrieves from the corpus the set of instances I that match any of the patterns in P. In Section 3.1.4, we propose a principled measure of instance reliability r_ι for ranking instances. Next, *Espresso* filters incorrect instances using the algorithm proposed in Section 3.2 and then selects the highest scoring m instances according to r_ι as input for the subsequent iteration. We experimentally set $m = 200$.

In small corpora, the number of extracted instances can be too low to guarantee sufficient statistical evidence for the pattern discovery phase of the next iteration. In such cases, the system enters an *expansion phase*, where instances are expanded as follows.

Web expansion: New instances of the patterns in P are retrieved from the Web, using the Google search engine. Specifically, for each instance $\{x, y\} \in I$, the system creates a set of queries, using each pattern in P instantiated with y. For example, given the instance *"Italy, country"* and the pattern *"Y such as X"*, the resulting Google query will be *"country such as *"*. New instances are then created from the retrieved Web results (e.g. *"Canada, country"*) and added to I. The noise generated from this expansion is attenuated by the filtering algorithm described in Section 3.2.

Syntactic expansion: New instances are created from each instance $\{x, y\} \in I$ by extracting sub-terminological expressions from x corresponding to the syntactic head of terms. For example, the relation *"new record of a criminal conviction part-of FBI report"* expands to: *"new record part-of FBI report"*, and *"record part-of FBI report"*.

3.1.4. Pattern and Instance Reliability

Intuitively, a reliable pattern is one that is both highly precise and one that extracts many instances. The recall of a pattern p can be approximated by the fraction of input instances that are extracted by p. Since it is non-trivial to estimate automatically the precision of a pattern, we are wary of keeping patterns that generate many instances (i.e., patterns that generate high recall but potentially disastrous precision). Hence, we desire patterns that are highly associated with the input instances. Pointwise mutual information [34] is a commonly used metric for measuring this strength of association between two events x and y:

$$pmi(x, y) = log \frac{P(x, y)}{P(x)P(y)}$$

We define the reliability $r_\pi(p)$ of a pattern p, as its average strength of association across each input instance $i \in I$, weighted by the reliability of each instance i:

$$r_\pi(p) = \frac{\sum_{i \in I} \frac{pmi(i, p)}{max_{pmi}} * r_\iota(i)}{|I|}$$

where $r_\iota(i)$ is the reliability of instance i (defined below) and max_{pmi} is the maximum pointwise mutual information between all patterns and all instances. $r_\pi(p)$ ranges from $[0, 1]$. The reliability of the manually supplied seed instances is $r_\iota(i) = 1$. The pointwise mutual information between instance $i = \{x, y\}$ and pattern p is estimated using the following formula:

$$pmi(i, p) = log \frac{|x, p, y|}{|x, *, y||*, p *|}$$

where $|x, p, y|$ is the frequency of pattern p instantiated with terms x and y and where the asterisk (*) represents a wildcard. A well-known problem is that pointwise mutual information is biased towards infrequent events. We thus multiply $pmi(i, p)$ with the discounting factor suggested by Pantel and Ravichandran [18].

Estimating the reliability of an instance is similar to estimating the reliability of a pattern. Intuitively, a reliable instance is one that is highly associated with as many reliable patterns as possible (i.e., we have more confidence in an instance when multiple reliable patterns instantiate it). Hence, analogous to our pattern reliability measure, we define the reliability $r_\iota(i)$ of an instance i as:

$$r_\iota(i) = \frac{\sum_{p \in P'} \frac{pmi(i, p)}{max_{pmi}} * r_\pi(p)}{|P'|}$$

where $r_\pi(p)$ is the reliability of pattern p (defined earlier) and max_{pmi} is as before. Note that $r_\iota(i)$ and $r_\pi(p)$ are recursively defined, where $r_\iota(i) = 1$ for the manually supplied seed instances.

3.2. Exploiting Generic Patterns

Generic patterns are high recall / low precision patterns (e.g, the pattern *"X of Y"* can ambiguously refer to a *part-of*, *is-a* and *possession* relations). Using them blindly increases system recall while dramatically reducing precision. Minimally supervised algorithms have typically ignored them for this reason. Only heavily supervised approaches, like Girju et Al. [15] have successfully exploited them.

Espresso's recall can be significantly increased by automatically separating correct instances extracted by generic patterns from incorrect ones. The challenge is to harness the expressive power of the generic patterns while remaining minimally supervised.

The intuition behind our method is that in a very large corpus, like the Web, correct instances of a generic pattern will be instantiated by many of *Espresso*'s reliable patterns accepted in P. Recall that, by definition, *Espresso*'s reliable patterns extract instances with high precision (yet often low recall). In a very large corpus, like the Web, we assume that a correct instance will occur in at least one of *Espresso*'s reliable pattern even though the patterns' recall is low. Intuitively, our confidence in a correct instance increases when, i) the instance is associated with many reliable patterns; and ii) its association with the reliable patterns is high. At a given Espresso iteration, where P_R represents the set of previously selected reliable patterns, this intuition is captured by the following measure of confidence in an instance $i = \{x, y\}$:

$$S(i) = \sum_{p \in P_R} S_p(i) \times \frac{r_\pi(p)}{T}$$

where T is the sum of the reliability scores $r_\pi(p)$ for each pattern $p \in P_R$, and

$$S_p(i) = pmi(i, p) = log \frac{|x, p, y|}{|x, *, y| \times |*, p*|}$$

where pointwise mutual information between instance i and pattern p is estimated with Google as follows:

$$S_p(i) \approx \frac{|x, p, y|}{|x| \times |y| \times |p|}$$

An instance i is rejected if $S(i)$ is smaller than some threshold τ. Although this filtering may also be applied to reliable patterns, we found this to be detrimental in our experiments since most instances generated by reliable patterns are correct. In *Espresso*, we classify a pattern as generic when it generates more than 10 times the instances of previously accepted reliable patterns.

4. Ontologizing Semantic Relations

The output of most relation harvesting algorithms, such as *Espresso* described in Section 3, consists of flat lists of lexical semantic knowledge such as *"Italy is-a country"* and *"orange similar-to blue"*. However, using this knowledge beyond simple keyword matching, for example in inferences, requires it to be linked, or *ontologized*, into semantic repositories such as ontologies or term banks like WordNet.

Given an instance (x, r, y) of a binary relation r between terms x and y, the ontologizing task is to identify the senses of x and y where r holds. In this work, we focus on WordNet 2.0 senses, though any similar term bank would apply.

Let S_x and S_y be the sets of all WordNet senses of x and y. A *sense pair*, s_{xy}, is defined as any pair of senses of x and y: $s_{xy} = \{s_x, s_y\}$ where $s_x \in S_x$ and $s_y \in S_y$. The set of all sense pairs S_{xy} consists of all pairings between senses in S_x and S_y.

In order to attach a relation instance (x, r, y) into WordNet, one must:

- *Disambiguate* x and y, that is, find the subsets $S'_x \subseteq S_x$ and $S'_y \subseteq S_y$ for which the relation r holds; and
- *Instantiate* the relation in WordNet, using the synsets corresponding to all correct pairings between the senses in S'_x and S'_y. We denote this set of attachment points as S'_{xy}.

If S_x or S_y is empty, no attachments are produced.

For example, the instance *(study, PART-OF, report)* is ontologized into WordNet through the senses $S'_x = \{survey\#1, study\#2\}$ and $S'_y = \{report\#1\}$. The final attachment points S'_{xy} are:

(survey#1, PART-OF, report#1)
(study#2, PART-OF, report#1)

Unlike common algorithms for word sense disambiguation, here it is important to take into consideration the semantic dependency between the two terms x and y. For example, an entity that is *part-of* a study has to be some kind of information. This knowledge about mutual selectional preference (the preferred semantic class that fills a certain relation role, as x or y) can be exploited to ontologize the instance.

In the following sections, we propose two algorithms for ontologizing binary semantic relations.

4.1. Method 1: Anchor Approach

Given an instance (x, r, y), this approach fixes the term y, called the anchor, and then disambiguates x by looking at all other terms that occur in the relation r with y. Based on the principle of distributional similarity [27], the algorithm assumes that the words that occur in the same relation r with y will be more similar to the correct sense(s) of x than the incorrect ones. After disambiguating x, the process is then inverted with x as the anchor to disambiguate y.

In the first step, y is fixed and the algorithm retrieves the set of all other terms X' that occur in an instance (x', r, y), $x' \in X'$[1]. For example, given the instance *(reflections, PART-OF, book)*, and a resource containing the following relations:

(false allegations, PART-OF, book)
(stories, PART-OF, book)
(expert analysis, PART-OF, book)
(conclusions, PART-OF, book)

[1]For semantic relations between complex terms, like *(expert analysis, PART-OF, book)*, only the head noun of terms are recorded, like *"analysis"*. As a future work, we plan to use the whole term if it is present in WordNet.

the resulting set X' would be: *{allegations, stories, analysis, conclusions}*. All possible pairings, $S_{xx'}$, between the senses of x and the senses of each term in X', called $S_{x'}$, are computed. For each sense pair $\{s_x, s_{x'}\}$ in $S_{xx'}$, a similarity score $r(s_x, s_{x'})$ is calculated using WordNet:

$$r(s_x, s_{x'}) = \frac{1}{d(s_x, s_{x'}) + 1} \times f(s_{x'})$$

where the distance $d(s_x, s_{x'})$ is the length of the shortest path connecting the two synsets in the hypernymy hierarchy of WordNet, and $f(s_{x'})$ is the number of times sense $s_{x'}$ occurs in any of the instances of X'. Note that if no connection between two synsets exists, then $r(s_x, s_{x'}) = 0$. The overall sense score for each sense s_x of x is calculated as:

$$r(s_x) = \sum_{s_{x'} \in S_{x'}} r(s_x, s_{x'})$$

Finally, the algorithm inverts the process by setting x as the anchor and computes $r(s_y)$ for each sense of y. All possible pairings of senses are computed and scored by averaging $r(s_x)$ and $r(s_y)$. Pairings scoring higher than a threshold τ_1 are selected as the attachment points in WordNet. We experimentally set $\tau_1 = 0.02$.

4.2. Method 2: Clustering Approach

The main idea of the clustering approach is to leverage the lexical behaviors of the two terms in an instance as a whole. The assumption is that the general meaning of the relation is derived from the combination of the two terms.

The algorithm is divided in two main phases. In the first phase, *semantic clusters* are built using the WordNet senses of all instances. A semantic cluster is defined by the set of instances that have a common semantic generalization. We denote the *conceptual instance* of the semantic cluster as the pair of WordNet synsets that represents this generalization. For example the following two *part-of* instances:

(second section, PART-OF, Los Angeles-area news)
(Sandag study, PART-OF, report)

are in a common cluster represented by the following conceptual instance:

[writing#2, PART-OF, message#2]

since *writing#2* is a hypernym of both *section* and *study*, and *message#2* is a hypernym of *news* and *report*[2].

In the second phase, the algorithm attaches an instance into WordNet by using WordNet distance metrics and frequency scores to select the best cluster for each instance. A good cluster is one that:

- achieves a good trade-off between generality and specificity; and

[2]Again, here, we use the syntactic head of each term for generalization since we assume that it drives the meaning of the term itself.

- disambiguates among the senses of x and y using the other instances' senses as support.

For example, given the instance *(second section, PART-OF, Los Angeles-area news)* and the following conceptual instances:

[writing#2, PART-OF, message#2]
[object#1, PART-OF, message#2]
[writing#2, PART-OF, communication#2]
[social_group#1, PART-OF, broadcast#2]
[organization#, PART-OF, message#2]

the first conceptual instance should be scored highest since it is both not too generic nor too specific and is supported by the instance *(Sandag study, PART-OF, report)*, i.e., the conceptual instance subsumes both instances. The second and the third conceptual instances should be scored lower since they are too generic, while the last two should be scored lower since the sense for *section* and *news* are not supported by other instances. The system then outputs, for each instance, the set of sense pairs that are subsumed by the highest scoring conceptual instance. In the previous example:

(section#1, PART-OF, news#1)
(section#1, PART-OF, news#2)
(section#1, PART-OF, news#3)

are selected, as they are subsumed by *[writing#2, PART-OF, message#2]*. These sense pairs are then retained as attachment points into WordNet.

Below, we describe each phase in more detail.

4.2.1. Phase 1: Cluster Building

Given an instance (x, r, y), all sense pair pairings $s_{xy} = \{s_x, s_y\}$ are retrieved from WordNet. A set of *candidate conceptual instances*, C_{xy}, is formed for each instance from the pairing of each WordNet ancestor of s_x and s_y, following the hypernymy link, up to degree τ_2.

Each candidate conceptual instance, $c = \{c_x, c_y\}$, is scored by its degree of generalization as follows:

$$r(c) = \frac{1}{(n_x + 1) \times (n_y + 1)}$$

where n_i is the number of hypernymy links needed to go from s_i to c_i, for $i \in \{x, y\}$. $r(c)$ ranges from $[0, 1]$ and is highest when little generalization is needed.

For example, the instance *(Sandag study, PART-OF, report)* produces 70 sense pairs since *study* has 10 senses and *report* has 7 senses. Assuming $\tau_2 = 1$, the instance sense *(survey#1, PART-OF, report#1)* has the set of candidate conceptual instances reported in Table 1.

Finally, each candidate conceptual instance c forms a cluster of all instances (x, r, y) that have some sense pair s_x and s_y as hyponyms of c. Note also that candidate conceptual instances may be subsumed by other candidate conceptual instances. Let G_c refer to the set of all candidate conceptual instances subsumed by candidate conceptual instance c.

C_{xy}	n_x	n_y	$r(c)$
(survey#1, PART-OF,report#1)	0	0	1
(survey#1, PART-OF,document#1)	0	1	0.5
(examination#1, PART-OF,report#1)	1	0	0.5
(examination#1, PART-OF,document#1)	1	1	0.25

Table 1. Example of conceptual instances for the instance sense *(survey#1, PART-OF, report#1)*.

Intuitively, better candidate conceptual instances are those that subsume both many instances and other candidate conceptual instances, but at the same time that have the least distance from subsumed instances. We capture this intuition with the following score of c:

$$score(c) = \frac{\sum\limits_{g \in G_c} r(g)}{|G_c|} \times log|I_C| \times log|G_C|$$

where I_c is the set of instances subsumed by c. We experimented with different variations of this score and found that it is important to put more weight on the distance between subsumed conceptual instances than the actual number of subsumed instances. Without the log terms, the highest scoring conceptual instances are too generic (i.e., they are too high up in the ontology).

4.2.2. Phase 2: Attachment Points Selection

In this phase, we utilize the conceptual instances of the previous phase to attach each instance (x, r, y) into WordNet.

At the end of Phase 1, an instance can be clustered in different conceptual instances. In order to select an attachment, the algorithm selects the sense pair of x and y that is subsumed by the highest scoring candidate conceptual instance. It and all other sense pairs that are subsumed by this conceptual instance are then retained as the final attachment points.

As a side effect, a final set of conceptual instances is obtained by deleting from each candidate those instances that are subsumed by a higher scoring conceptual instance. Remaining conceptual instances are then rescored using $score(c)$. The final set of conceptual instances thus contains unambiguous sense pairs.

5. Experimental Results

In this section, we evaluate both the harvesting algorithm *Espresso* and our two algorithms for ontologizing semantic relations.

5.1. Espresso Evaluation

Here, we present an empirical comparison of *Espresso* with three state of the art systems on the task of extracting various semantic relations.

5.1.1. Experimental Setup

We perform our experiments using the following two datasets:

- **TREC**: This dataset consists of a sample of articles from the Aquaint (TREC-9) newswire text collection. The sample consists of 5,951,432 words extracted from the following data files: AP890101 - AP890131, AP890201 - AP890228, and AP890310 - AP890319.
- **CHEM**: This small dataset of 313,590 words consists of a college level textbook of introductory chemistry [35].

Each corpus is pre-processed using the Alembic Workbench POS-tagger [36].

Below we describe the systems used in our empirical evaluation of *Espresso*.

- **RH02**: The algorithm by Ravichandran and Hovy [14] described in Section 2.
- **GI06**: The algorithm by Girju et Al. [15] described in Section 2.
- **PR04**: The algorithm by Pantel [18] described in Section 2.
- **ESP-**: The *Espresso* algorithm using the pattern and instance reliability measures, but without using generic patterns.
- **ESP+**: The full *Espresso* algorithm described in this work exploiting generic patterns.

For *ESP*+, we experimentally set τ from Section 3.2 to $\tau = 0.4$ for TREC, and $\tau = 0.3$ for CHEM by manually inspecting a small set of instances.

Espresso is designed to extract various semantic relations exemplified by a given small set of seed instances. We consider the standard *is-a* and *part-of* relations as well as the following more specific relations:

- *succession*. This relation indicates that a person succeeds another in a position or title. For example, *George Bush* succeeded *Bill Clinton* and *Pope Benedict XVI* succeeded *Pope John Paul II*. We evaluate this relation on the TREC-9 corpus.
- *reaction*. This relation occurs between chemical elements/molecules that can be combined in a chemical reaction. For example, *hydrogen gas* reacts with *oxygen gas* and *zinc* reacts-with *hydrochloric acid*. We evaluate this relation on the CHEM corpus.
- *production*. This relation occurs when a process or element/object produces a result[3]. For example, *ammonia* produces *nitric oxide*. We evaluate this relation on the CHEM corpus.

For each semantic relation, we manually extracted a small set of seed examples. The seeds were used for both *Espresso* as well as *RH02*. Table 2 lists a sample of the seeds as well as sample outputs from *Espresso*.

5.1.2. Precision and Recall

We implemented the systems outlined in Section 5.1.1, except for *GI06*, and applied them to the TREC and CHEM datasets. For each output set, per relation, we evaluate the precision of the system by extracting a random sample of instances (50 for the TREC corpus and 20 for the CHEM corpus) and evaluating their quality manually using two

[3] *Production* is an ambiguous relation; it is intended to be a causation relation in the context of chemical reactions.

	Is-a (12)	*Part-Of (12)*	*Succession (12)*	*Reaction (13)*	*Production (14)*
Seeds	wheat :: crop	leader :: panel	Khrushchev :: Stalin	magnesium :: oxygen	bright flame :: flares
	George Wendt :: star	city :: region	Carla Hills :: Yeutter	hydrazine :: water	hydrogen :: metal hydrides
	nitrogen :: element	ion :: matter	Bush :: Reagan	aluminum metal :: oxygen	ammonia :: nitric oxide
	diborane :: substance	oxygen :: water	Julio Barbosa :: Mendes	lithium metal :: fluorine gas	copper :: brown gas
Espresso	Picasso :: artist	trees :: land	Ford :: Nixon	hydrogen :: oxygen	electron :: ions
	tax :: charge	material :: FBI report	Setrakian :: John Griesemer	Ni :: HCl	glycerin :: nitroglycerin
	protein :: biopolymer	oxygen :: air	Camero Cardiel :: Camacho	carbon dioxide :: methane	kidneys :: kidney stones
	HCl :: strong acid	atom :: molecule	Susan Weiss :: editor	boron :: fluorine	ions :: charge

Table 2. Sample seeds used for each semantic relation and sample outputs from *Espresso*. The number in the parentheses for each relation denotes the total number of seeds used as input for the system.

human judges (a total of 680 instances were annotated per judge). For each instance, judges may assign a score of 1 for correct, 0 for incorrect, and 1/2 for partially correct. Example instances that were judged partially correct include *"analyst is-a manager"* and *"pilot is-a teacher"*. The kappa statistic [37] on this task was $K = 0.69$[4]. The precision for a given set of instances is the sum of the judges' scores divided by the total instances.

Although knowing the total number of correct instances of a particular relation in any non-trivial corpus is impossible, it is possible to compute the recall of a system relative to another system's recall. Following Pantel et Al. [21], we define the relative recall of system A given system B, $R_{A|B}$, as:

$$R_{A|B} = \frac{R_A}{R_B} = \frac{\frac{C_A}{C}}{\frac{C_B}{C}} = \frac{C_A}{C_B} = \frac{P_A \times |A|}{P_B \times |B|}$$

where R_A is the recall of A, C_A is the number of correct instances extracted by A, C is the (unknown) total number of correct instances in the corpus, P_A is A's precision in our experiments, and $|A|$ is the total number of instances discovered by A.

Tables 3-9 report the total number of instances, precision[5], and relative recall[6] of each system on the TREC-9 and CHEM corpora. The relative recall is always given in relation to the *ESP-* system. For example, in Table 3, *RH02* has a relative recall of 5.31 with *ESP-*, which means that the *RH02* system outputs 5.31 times more correct relations than *ESP-* (at a cost of much lower precision). Similarly, *PR04* has a relative recall of 0.23 with *ESP-*, which means that *PR04* outputs 4.35 fewer correct relations than *ESP-* (also with a smaller precision). We did not include the results from *GI06* in the tables since the system is only applicable to *part-of* relations and we did not reproduce it. However, the authors evaluated their system on a sample of the TREC-9 dataset and reported 83% precision and 72% recall (this algorithm is heavily supervised.)

In all tables, *RH02* extracts many more relations than *ESP-*, but with a much lower precision, because it uses generic patterns without filtering. The high precision of *ESP-* is due to the effective reliability measures presented in Section 3.1.4.

5.1.3. *Effect of Generic Patterns*

Experimental results, for all relations and the two different corpus sizes, show that *ESP-* greatly outperforms the other methods on precision. However, without the use of generic patterns, the *ESP-* system shows lower recall in all but the *production* relation.

[4]The kappa statistic jumps to K = 0.79 if we treat partially correct classifications as correct.

[5]Because of the small evaluation sets, we estimate the 95% confidence intervals using bootstrap resampling to be in the order of $\pm$ 10-15% (absolute numbers).

[6]Relative recall is given in relation to ESP-.

SYSTEM	INSTANCES	PRECISION	REL RECALL
RH02	57,525	28.0%	5.31
PR04	1,504	47.0%	0.23
ESP-	4,154	**73%**	1.00
ESP+	69,156	36.2%	**8.26**

Table 3. System performance: TREC/*is-a*.

SYSTEM	INSTANCES	PRECISION	REL RECALL
RH02	2556	25.0%	3.76
PR04	108	40.0%	0.25
ESP-	200	**85.0%**	1.00
ESP+	1490	76.0%	**6.66**

Table 4. System performance: CHEM/*is-a*.

SYSTEM	INSTANCES	PRECISION	REL RECALL
RH02	12,828	35.0%	42.52
ESP-	132	**80.0%**	1.00
ESP+	87,203	69.9%	**577.22**

Table 5. System performance: TREC/*part-of*.

SYSTEM	INSTANCES	PRECISION	REL RECALL
RH02	11,582	33.8%	**58.78**
ESP-	111	**60.0%**	1.00
ESP+	5973	50.7%	45.47

Table 6. System performance: CHEM/*part-of*.

SYSTEM	INSTANCES	PRECISION	REL RECALL
RH02	49,798	2.0%	**36.96**
ESP-	55	**49.0%**	1.00
ESP+	55	**49.0%**	1.00

Table 7. System performance: TREC/*succession*.

SYSTEM	INSTANCES	PRECISION	REL RECALL
RH02	6,083	30%	53.67
ESP-	40	85%	1.00
ESP+	3102	**91.4%**	**89.39**

Table 8. System performance: CHEM/*reaction*.

SYSTEM	INSTANCES	PRECISION	REL RECALL
RH02	197	57.5%	0.80
ESP-	196	**72.5%**	1.00
ESP+	1676	55.8%	**6.58**

Table 9. System performance: CHEM/*production*.

As hypothesized, exploiting generic patterns using the algorithm from Section 3.2 substantially improves recall without much deterioration in precision. *ESP+* shows one to two orders of magnitude improvement on recall while losing on average below 10% precision. The *succession* relation in Table 7 was the only relation where *Espresso* found no generic pattern. For other relations, *Espresso* found from one to five generic patterns. Table 5 shows the power of generic patterns where system recall increases by 577 times with only a 10% drop in precision. In Table 8, we see a case where the combination of filtering with a large increase in retrieved instances resulted in both higher precision and recall.

In order to better analyze our use of generic patterns, we performed the following experiment. For each relation, we randomly sampled 100 instances for each generic pattern and built a gold standard for them (by manually tagging each instance as correct or incorrect). We then sorted the 100 instances according to the scoring formula $S(i)$ derived in Section 3.2 and computed the average precision, recall, and F-score of each top-K ranked instances for each pattern[7]. Due to lack of space, we only present the graphs for four of the 22 generic patterns: *"X is a Y"* for the is-a relation of Table 3, *"X in the Y"* for the *part-of* relation of Table 5, *"X in Y"* for the *part-of* relation of Table 6, and *"X and Y"* for the reaction relation of Table 8. Figure 1 illustrates the results.

In each figure, notice that recall climbs at a much faster rate than precision decreases. This indicates that the scoring function of Section 3.2 effectively separates correct and

[7]We can directly compute recall here since we built a gold standard for each set of 100 samples.

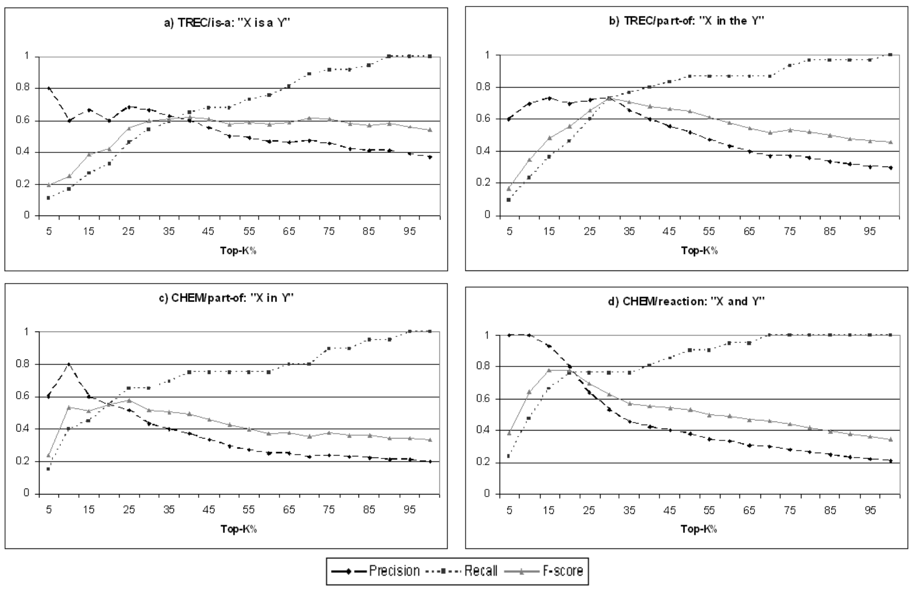

Figure 1. Precision, recall and *F*-score curves of the Top-*K%* ranking instances of patterns *"X is a Y"* (TREC/*is-a*), *"X in Y"* (TREC/*part-of*), *"X in the Y"* (CHEM/*part-of*), and *"X and Y"* (CHEM/*reaction*).

incorrect instances. In Figure 1.a), there is a big initial drop in precision that accounts for the poor precision reported in Table 3.

Recall that the cutoff points on $S(i)$ were set to $\tau = 0.4$ for TREC and $\tau = 0.3$ for CHEM. The figures show that this cutoff is far from the maximum *F*-score. An interesting avenue of future work would be to automatically determine the proper threshold for each individual generic pattern instead of setting a uniform threshold.

5.2. *Ontologizing Evaluation*

Here, we present an empirical evaluation of our two methods for ontologizing binary semantic relations, presented in Section 4.

5.2.1. *Experimental Setup*

Researchers have developed many algorithms for harvesting semantic relations from corpora and the Web. For the purposes of this work, we may choose any one of them and manually validate its mined relations. We choose *Espresso*, the harvesting algorithm described in Section 3.

Test Sets

We experiment with two relations: *part-of* and *causation*. The *causation* relation occurs when an entity produces an effect or is responsible for events or results, for example *(virus, CAUSE, influenza)* and *(burning fuel, CAUSE, pollution)*. We manually built five seed relation instances for both relations and apply *Espresso* to a dataset consisting of a sample of articles from the Aquaint (TREC-9) newswire text collection. The sample consists of 55.7 million words extracted from the Los Angeles Times data files. *Espresso* extracted 1,468 *part-of* instances and 1,129 *causation* instances. We manually validated the output and randomly selected 200 correct relation instances of each relation for ontologizing into WordNet 2.0.

Gold Standard

We manually built a gold standard of all correct attachments of the test sets in WordNet. For each relation instance (x, r, y), two human annotators selected from all sense pairings of x and y the correct attachment points in WordNet. For example, for *(synthetic material, PART-OF, filter)*, the judges selected the following attachment points: *(synthetic material#1, PART-OF, filter#1)* and *(synthetic material#1, PART-OF, filter#2)*. The kappa statistic [37] on the two relations together was $K = 0.73$.

Systems

The following three systems are evaluated:

- **BL**: the baseline system that attaches each relation instance to the first (most common) Word-Net sense of both terms;
- **AN**: the anchor approach described in Section 4.1.
- **CL**: the clustering approach described in Section 4.2.

5.2.2. *Precision, Recall and* F-*score*

For both the *part-of* and *causation* relations, we apply the three systems described above and compare their attachment performance using precision, recall, and *F*-score. Using the manually built gold standard, the precision of a system on a given relation instance is measured as the percentage of correct attachments and recall is measured as the percentage of correct attachments retrieved by the system. Overall system precision and recall are then computed by averaging the precision and recall of each relation instance.

Table 10 and Table 11 report the results on the *part-of* and *causation* relations. We experimentally set the CL generalization parameter τ_2 to 5 and the τ_1 parameter for AN to 0.02.

SYSTEM	PRECISION	RECALL	F-SCORE
BL	54.0%	31.3%	39.6%
AN	40.7%	47.3%	43.8%
CL	57.4%	49.6%	53.2%

Table 10. System performance on the *part-of* relation.

SYSTEM	PRECISION	RECALL	F-SCORE
BL	45.0%	25.0%	32.1%
AN	41.7%	32.4%	36.5%
CL	40.0%	32.6%	35.9%

Table 11. System performance on the *causation* relation.

5.2.3. *Discussion*

For both relations, CL and AN outperform the baseline in overall *F*-score. For *part-of*, Table 10 shows that CL outperforms BL by 13.6% in *F*-score and AN by 9.4%. For causation, Table 11 shows that AN outperforms BL by 4.4% on *F*-score and CL by 0.6%.

The good results of the CL method on the *part-of* relation suggest that instances of this relation are particularly amenable to be clustered. The generality of the *part-of* relation in fact allows the creation of fairly natural clusters, corresponding to different sub-types of *part-of*, as those proposed by Winston et Al. [38]. The *causation* relation, however, being more difficult to define at a semantic level [39], is less easy to cluster and thus to disambiguate.

Both CL and AN have better recall than BL, but precision results vary with CL beating BL only on the *part-of* relation. Overall, the system performances suggest that ontologizing semantic relations into WordNet is in general not easy.

The better results of CL and AN with respect to BL suggest that the use of comparative semantic analysis among corpus instances is a good way to carry out disambiguation. Yet, the BL method shows surprisingly good results. This indicates that also a simple method based on word sense usage in language can be valuable. An interesting avenue of future work is to better combine these two different views in a single system.

The low recall results for CL are mostly attributed to the fact that in Phase 2 only the best scoring cluster is retained for each instance. This means that instances with multiple senses that do not have a common generalization are not captured. For example the *part-of* instance *(wings, PART-OF, chicken)* should cluster both in *[body_part#1, PART-OF, animal#1]* and *[body_part#1, PART-OF, food#2]*, but only the best scoring one is retained.

5.2.4. Conceptual Instances: Other Uses

Our *clustering approach* from section 4.2 is enabled by learning conceptual instances - relations between mid-level ontological concepts. Beyond the ontologizing task, conceptual instances may be useful for several other tasks. In this Section, we discuss some of these opportunities and present small qualitative evaluations.

Conceptual instances represent common semantic generalizations of a particular relation. For example, below are two possible conceptual instances for the *part-of* relation:

[person#1, PART-OF, organization#1]
[act#1, PART-OF, plan#1]

The first conceptual instance in the example subsumes all the *part-of* instances in which one or more persons are part of an organization, such as:

(president Brown, PART-OF, executive council)
(representatives, PART-OF, organization)
(students, PART-OF, orchestra)
(players, PART-OF, Metro League)

Below, we present three possible ways of exploiting these conceptual instances.

Support to Relation Extraction Tools

Conceptual instances may be used to support relation extraction algorithms such as *Espresso*.

Most minimally supervised harvesting algorithm do not exploit *generic patterns*, i.e. those patterns with high recall but low precision, since they cannot separate correct and incorrect relation instances. For example, the pattern *"X of Y"* extracts many correct relation instances like *"wheel of the car"* but also many incorrect ones like *"house of representatives"*.

Girju et Al. [15] described a highly supervised algorithm for learning semantic constraints on *generic patterns*, leading to a very significant increase in system recall with-

CONCEPTUAL INSTANCES	SCORE	# INSTANCES	INSTANCES
[multitude#3, PART-OF, group#1]	2.04	10	(ordinary people, PART-OF, Democratic Revolutionary Party) (unlicensed people, PART-OF, underground economy) (young people, PART-OF, commission) (air mass, PART-OF, cold front)
[person#1, PART-OF, organization#1]	1.71	43	(foreign ministers, PART-OF, council) (students, PART-OF, orchestra) (socialists, PART-OF, Iraqi National Joint Action Committee) (players, PART-OF, Metro League)
[act#2, PART-OF, plan#1]	1.60	16	(major concessions, PART-OF, new plan) (attacks, PART-OF, coordinated terrorist plan) (visit, PART-OF, exchange program) (survey, PART-OF, project)
[communication#2, PART-OF, book#1]	1.14	10	(hints, PART-OF, booklet) (soup recipes, PART-OF, book) (information, PART-OF, instruction manual) (extensive expert analysis, PART-OF, book)
[compound#2, PART-OF, waste#1]	0.57	3	(salts, PART-OF, powdery white waste) (lime, PART-OF, powdery white waste) (resin, PART-OF, waste)

Table 12. Sample of the highest scoring conceptual instances learned for the *part-of* relation. For each conceptual instance, we report score(c) , the number of instances, and some example instances.

CONCEPTUAL INSTANCES	SCORE	# INSTANCES	INSTANCES
[change#3, CAUSE, state#4]	1.49	17	(separation, CAUSE, anxiety) (demotion, CAUSE, roster vacancy) (budget cuts, CAUSE, enrollment declines) (reduced flow, CAUSE, vacuum)
[act#2, CAUSE, state#3]	0.81	20	(oil drilling, CAUSE, air pollution) (workplace exposure, CAUSE, genetic injury) (industrial emissions, CAUSE, air pollution) (long recovery, CAUSE, great stress)
[person#1, CAUSE, act#2]	0.64	12	(homeowners, CAUSE, water waste) (needlelike puncture, CAUSE, physician) (group member, CAUSE, controversy) (children, CAUSE, property damage)
[organism#1, CAUSE, disease#1]	0.03	4	(parasites, CAUSE, pneumonia) (virus, CAUSE, influenza) (chemical agents, CAUSE, pneumonia) (genetic mutation, CAUSE, Dwarfism)

Table 13. Sample of the highest scoring conceptual instances learned for the *causation* relation. For each conceptual instance, we report score(c) , the number of instances, and some example instances.

out deteriorating precision. Conceptual instances can be used to automatically learn such semantic constraints by acting as a filter for generic patterns, retaining only those instances that are subsumed by high scoring conceptual instances. Effectively, conceptual instances are used as *selectional restrictions* for the relation. For example, our system discards the following incorrect instances:

(week, CAUSE, coalition)
(demeanor, CAUSE, vacuum)

as they are both part of the very low scoring conceptual instance *[abstraction#6, CAUSE, state#1]*.

Ontology Learning from Text

Each conceptual instance can be viewed as a formal specification of the relation at hand.

For example, Winston et Al. [38] manually identified six sub-types of the *part-of* relation: *member-collection*, *component-integral object*, *portion-mass*, *stuff-object*, *feature-activity* and *place-area*. Such classifications are useful in applications and tasks where a semantically rich organization of knowledge is required. Conceptual instances can be viewed as an automatic derivation of such a classification based on corpus usage. Moreover, conceptual instances can be used to improve the ontology learning process itself. For example, our *clustering approach* can be seen as an *inductive* step producing conceptual instances that are then used in a *deductive* step to learn new instances. An algorithm could iterate between the induction/deduction cycle until no new relation instances and conceptual instances can be inferred.

Word Sense Disambiguation
Word Sense Disambiguation (WSD) systems can exploit the selectional restrictions identified by conceptual instances to disambiguate ambiguous terms occurring in particular contexts. For example, given the sentence:

"the board is composed by members of different countries"

and a harvesting algorithm that extracts the *part-of* relation *(members, PART-OF, board)*, the system could infer the correct senses for board and members by looking at their closest conceptual instance. In our system, we would infer the attachment *(member#1, PART-OF, board#1)* since it is part of the highest scoring conceptual instance *[person#1, PART-OF, organization#1]*.

Qualitative Evaluation
Table 12 and Table 13 list samples of the highest ranking conceptual instances obtained by our system for the *part-of* and *causation* relations. Below we provide a small evaluation to verify:

- the *correctness* of the conceptual instances. Incorrect conceptual instances such as *[attribute#2, CAUSE, state#4]*, discovered by our system, can impede WSD and extraction tools where precise selectional restrictions are needed; and
- the *accuracy* of the conceptual instances. Sometimes, an instance is incorrectly attached to a correct conceptual instance. For example, the instance *(air mass, PART-OF, cold front)* is incorrectly clustered in *[group#1, PART-OF, multitude#3]* since mass and front both have a sense that is descendant of *group#1* and *multitude#3*. However, these are not the correct senses of *mass* and *front* for which the *part-of* relation holds.

For evaluating *correctness*, we manually verify how many correct conceptual instances are produced by Phase 2 of the clustering approach described in Section 4.2. The claim is that a *correct* conceptual instance is one for which the relation holds for all possible subsumed senses. For example, the conceptual instance *[group#1, PART-OF, multitude#3]* is correct, as the relation holds for every semantic subsumption of the two senses. An example of an incorrect conceptual instance is *[state#4, CAUSE, abstraction#6]* since it subsumes the incorrect instance *(audience, CAUSE, new context)*. A manual evaluation of the highest scoring 200 conceptual instances, generated on our test sets described in Section 5.2.1, showed 82% correctness for the *part-of* relation and 86% for *causation*.

For estimating the overall clustering *accuracy*, we evaluated the number of correctly clustered instances in each conceptual instance. For example, the instance *(business people, PART-OF, committee)* is correctly clustered in *[multitude#3, PART-OF, group#1]* and the instance *(law, PART-OF, constitutional pitfalls)* is incorrectly clustered in *[group#1, PART-OF, artifact#1]*. We estimated the overall accuracy by manually judging the instances attached to 10 randomly sampled conceptual instances. The accuracy for *part-of* is 84% and for causation it is 76.6%.

6. Conclusions

In this chapter, we presented algorithms for both extracting semantic relations from textual resources and for linking, or *ontologizing*, them into a semantic repository. We proposed a weakly-supervised, general-purpose, and accurate algorithm, called *Espresso*, for harvesting binary semantic relations from raw text. The main contributions are: i) a method for exploiting generic patterns by filtering incorrect instances using the Web; and ii) a principled measure of pattern and instance reliability enabling the filtering algorithm. We have empirically compared *Espresso*'s precision and recall with other systems on both a small domain-specific textbook and on a larger corpus of general news, and have extracted several standard and specific semantic relations: *is-a*, *part-of*, *succession*, *reaction*, and *production*. *Espresso* achieves higher and more balanced performance than other state of the art systems. By exploiting generic patterns, system recall substantially increases with little effect on precision.

We then proposed two algorithms for automatically ontologizing binary semantic relations into WordNet: an *anchoring approach* and a *clustering approach*. Experiments on the *part-of* and *causation* relations showed promising results. Both algorithms outperformed the baseline on F-score. Our best results were on the *part-of* relation where the clustering approach achieved 13.6% higher F-score than the baseline. The induction of conceptual instances has opened the way for many avenues of future work. We intend to pursue the ideas presented in Section 5.2.4 for using conceptual instances to: i) support knowledge acquisition tools by learning semantic constraints on extracting patterns; ii) support ontology learning from text; and iii) improve word sense disambiguation through selectional restrictions. Also, we will try different similarity score functions for both the clustering and the anchoring approaches, as those surveyed in Corley and Mihalcea [40].

The algorithms described in this chapter may be applied to ontologize many lexical resources of semantic relations, no matter the harvesting algorithm used to mine them. In doing so, we have the potential to quickly enrich our ontologies, like WordNet, thus reducing the knowledge acquisition bottleneck. It is our hope that we will be able to leverage these enriched resources, albeit with some noisy additions, to improve performance on knowledge rich problems such as question answering and information extraction.

References

[1] M. Pasca and S. Harabagiu. The informative role of wordnet in open-domain question answering. In *Proceedings of the NAACL-2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 138–143, Pittsburgh, PA, 2001.

[2] M. Geffet and I. Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, Ann Arbor, MI, 2005.

[3] K. Mahesh. Ontology development for machine translation: Ideology and methodology. Rl report mccs-96-292, New Mexico State University, 1996.

[4] Mahesh K. O'Hara, T and S. Nirenburg. Lexical acquisition with wordnet and the mikrokosmos ontology. In *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, Montreal, Canada, 1998.

[5] Guarino N. Gangemi, A., C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with dolce. In *Proceedings of Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002*, pages 166–181, Siguenza, Spain, 2002.

[6] I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, pages 2–9, Ogunquit, Maine, 2001.

[7] C. Baker, C. Fillmore, and J. Lowe. The berkeley framenet project. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-98)*, pages 86–90, Montreal, Canada, 1998.

[8] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[9] O. Etzioni, M.J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, (165(1)):91–134, 2005.

[10] E. Riloff and J. Shepherd. A corpus-based approach for building semantic lexicons. In *Proceedings of 2nd Conference on Empirical Methods in Natural Language Processing (EMNLP-2007))*, pages 117–124, Somerset, NJ, 1997.

[11] D. Lin and P. Pantel. Concept discovery from text. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-02)*, pages 577–583, Taipei, Taiwan, 2002.

[12] D. Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28rd Annual Meeting of the Association for Computational Linguistics (ACL-1990)*, pages 268–275, Pittsburgh, PA, 1990.

[13] H.; Dagan I.; Szpektor, I.; Tanev and B. Coppola. Scaling web-based acquisition of entailment relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 41–48, Barcelona, Spain, 2004.

[14] D. Ravichandran and E.H. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 41–47, Philadelphia, PA, 2002.

[15] R. Girju, A. Badulescu, and D. Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, (32(1)):83–135, 2006.

[16] J. Bos. Invited talk. In *2nd Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, Sydney, Australia, 2006. Association for Computational Linguistics.

[17] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545, Nantes, France, 1992.

[18] P. Pantel and D. Ravichandran. Automatically labeling semantic classes. In *Proceedings of Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-04)*, pages 321–328, Boston, MA, 2004.

[19] P. Pantel. Inducing ontological co-occurrence vectors. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, pages 125–132, Ann Arbor, MI, 2005.

[20] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL-1999)*, pages 57–64, College Park, MD, 1999.

[21] P. Pantel, D. Ravichandran, and E.H. Hovy. Towards terascale knowledge acquisition. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING-04)*, pages 771–777, Geneva, Switzerland, 2004.

[22] G. S. Mann. Fine-grained proper noun ontologies for question answering. In *Proceedings of SemaNet' 02: Building and Using Semantic Networks*, pages 1–7, Taipei, Taiwan, 2002.

[23] D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction.

In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1034–1041, Edinburgh, Scotland, 2005.

[24] R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of the 7th Neural Information Processing System Conference (NIPS-05)*, Vancouver, Canada, 2005.

[25] B. Roark and E. Charniak. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-98)*, pages 1110–1116, Montreal, Canada, 1998.

[26] S. Caraballo. Automatic acquisition of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-1999)*, pages 57–64, College Park, MD, 1999.

[27] Z. Harris. *Distributional structure*, pages 26–47. New York: Oxford University Press, 1985.

[28] R. Basili, M.T. Pazienza, and M. Vindigni. Corpus-driven learning of event recognition rules. In *Proceedings of Workshop on Machine Learning for Information Extraction workshop held in conjunction with the 14th European Conference on Artificial Intelligence (ECAI-00)*, Berlin, Germany, 2000.

[29] E. Agirre and G. Rigau. Word sense disambiguation using conceptual density. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 16–22, Copenhagen, Danmark, 1996.

[30] S. Harabagiu, G. Miller, and D. Moldovan. Wordnet 2 - a morphologically and semantically enhanced resource. In *Proceedings of SIGLEX-99*, pages 1–8, University of Maryland, 1999.

[31] E. Agirre, G. Rigau, D. Martinez, and E.H. Hovy. Enriching wordnet concepts with topic signatures. In *Proceedings of the NAACL-2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, Pittsburgh, PA, 2001.

[32] W. Gale, K. Church, and D. Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and Humanities*, (26):415–439, 1992.

[33] J. Justeson and S. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, (1):9–27, 1995.

[34] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley and Sons, 1991.

[35] T.L. Brown, H.E. LeMay, and B.E. Bursten. *Chemistry: The Central Science*. Prentice Hall, 2003.

[36] D. Day, J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, , and M. Vilain. Mixed-initiative development of language processing systems. In *Proceedings of Fifth Conference on Applied Natural Language Processing (ANLP-97)*, pages 348–355, Washington D.C., 1997.

[37] S. Siegel and N. J. Castellan Jr. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 1998.

[38] M. Winston, R. Chaffin, and D. Hermann. A taxonomy of part-whole relations. *Cognitive Science*, (11):417–444, 1987.

[39] R. Girju. Automatic detection of causal relations for question answering. In *Proceedings of ACL Workshop on Multilingual Summarization and Question Answering*, pages 107–114, Sapporo, Japan, 2003.

[40] C. Corley and R. Mihalcea. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modelling of Semantic Equivalence and Entailment*, pages 13–18, Ann Arbor, MI, 2005.