Customisable Semantic Analysis of Texts

Vivi Nastase and Stan Szpakowicz

School of Information Technology and Engineering,
University of Ottawa, Ottawa, Ontario, Canada
{vnastase, szpak}@site.uottawa.ca

Abstract. Our customisable semantic analysis system implements a form of knowledge acquisition. It automatically extracts syntactic units from a text and semi-automatically assigns semantic information to pairs of units. The user can select the type of units of interest and the list of semantic relations to be assigned. The system examines parse trees to decide if there is interaction between concepts that underlie syntactic units. Memory-based learning proposes the most likely semantic relation for each new pair of syntactic units that may be semantically linked. We experiment with several configurations, varying the syntactic analyzer and the list of semantic relations.

1 Introduction

Deep processing of natural language data often requires suitably annotated data. Recognition of semantic relations is such a task that benefits from the availability of annotated texts from which we can learn to analyze new data. Manual semantic annotation is a time-consuming activity, and it is seldom possible to capitalize on the annotation effort of other researchers. This is because they work with a different set of semantic phenomena, for example a different list of relations, or because they consider different types of texts or different domains. We present a customizable, domain-independent tool for certain style of semantic analysis. It relies on syntactic information usually supplied by parsing. When the tool achieves its full functionality, its user will be able to impose her own list of semantic relations, select the type of relations she is interested in (between eventsm between an event and an entity, and so on), and plug in her own parser.

Knowledge acquisition from texts spans the range between fully automatic and fully user-driven systems. Automation relies on manually built resources and on statistical or machine-learning methods that extract classifiers from annotated data. The shortcomings of such methods include high cost of annotation and low accuracy of such classifiers on new data. User-driven systems, with friendly interfaces that domain experts use to identify knowledge in texts, allow much higher accuracy (insofar as humans agree on semantic relations). On the other hand, they require time to train people with minimal AI or NLP background, and to encode knowledge.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 312–323, 2005.

Our approach falls between these extremes. We rely on parsers for the grammatical structure of sentences, in which we identify concepts and pair up those that may interact. The user will associate the types of concepts of interest with syntactic units that the parser's grammar recognizes. For example, if entities are sought, the user will choose nouns and noun-phrases.

Our system extracts pairs of syntactic units from the text, which express concepts that according to syntactic indicators are semantically linked. Each pair is assigned a semantic relation that describes their interaction in the context in which they appear. While there is a default list of 47 semantic relations, the actual list may be user-defined, to acknowledge the fact that no set of semantic relations is appropriate for all NLP tasks. Semantic relations are assigned to pairs semi-automatically. The user can accept a unique suggestion made by the system, choose from a (usually short) list, enter the correct answer manually or reject the pair.

Barker et al. [1] presented and tested a similar idea. One of our innovations is to treat the input text uniformly, without separating syntactic levels (noun phrase, simple clause, compound clause, paragraph and so on). This emphasizes the fact that the same concept can surface in different syntactic forms. We let the user decide what structures are interesting, and focus on the concepts behind these structures. We use syntactic clues to decide which structures interact and to label the interaction. The user may specify the list of semantic relations that best fit the domain and the application.

This paper is organized as follows. Section 2 presents related work in semantic analysis and knowledge acquisition, Section 3 describes the semantic analysis process used by our system, the experiments performed are presented in Section 4, and their results are discussed in Section 5; Section 6 assesses the system's customisability, and the conclusions are presented in Section 7.

2 Related Work

One style of semantic analysis for knowledge acquisition uses predefined templates, filled with information from processed texts [2]. In other systems lexical resources are specifically tailored to meet the requirements of the domain [3] or of the system [4]. Such systems extract information from some types of syntactic units: clauses [5], [6], [7] and noun-phrases [7], [8]. Lists of semantic relations are designed to capture salient information from the domain.

An interesting approach has been tested in the Rapid Knowledge Formation project. The goal was to develop a system for domain experts to build complex knowledge bases by combining components: events, entities and modifiers [9]. The system's interface facilitates the expert's task of creating and manipulating structures representing domain concepts. Descriptions of relations between components come from a relation dictionary; it includes interaction between two events (e.g., causality), an event and the entities involved (e.g., agent), an entity and an event (e.g., capability), two entities (e.g., part), or an event or entity and their properties (e.g., duration or size) [10]. The relations cover three syntactic levels [11].

In purely statistical approaches that traverse corpora to establish connections between concepts based on word collocations, the incidence of errors is not negligible [12], [13], [14].

In our system, user feedback helps produce accurate results, and we will extract knowledge tailored to the user's interests. The knowledge acquisition systems that we have considered suggest that in some domains relations between entities are considered more important, e.g., in medicine [3]. In others it is important to see how entities are related to an event, e.g., in legal texts [2]. We are building a *customisable* system that will focus on the structures of interest to a particular domain. We also experiment with two different lists of relations, to test the flexibility of the semantic analysis module. The goal is to allow the user to plug in a list of relations that describes the input text best.

3 Semantic Analysis

To get the grammatical structure of the input sentence we need a parser, preferably one that has good coverage and produces detailed syntactic information. The parse trees give us syntactic units, from which we choose those of interest to the user, based on the information he provides (explained in detail in Section 3.1). To pair units up we use simple structural information: if a unit is directly embedded in another unit, we assume a subordinate relation between the two; if the two units are coordinate, we assume a coordinate relation. These assumptions are safe if the parse is correct: a modifier is subordinate to its head noun, an argument to its head verb, and a clause perhaps to the main clause in the sentence. If we conclude that two units should interact, we seek an appropriate semantic relation to describe this interaction.

3.1 Extracting Syntactic Units

The user can specify a list of syntactic structures of interest among those recognized by the parser's grammar. It will contain the relevant non-terminals. For example, if the user is interested in entities and their attributes, the list will contain non-terminals that describe nouns, noun phrases and their modifiers. If the user is interested in events and the way they interact, the list will contain non-terminals that describe clauses in the grammar. To simplify the interaction, we let the user choose the corresponding syntactic level (noun phrase, intra-clause or clause level). To allow finer-grained distinctions we will construct a tool that helps the user make a detailed unit selection.

Each syntactic unit will be represented by the uninflected form of its head word. For each unit we also extract the head word's **part of speech**, the **syntactic role** it plays in the sentence (subject, object, noun modifier, etc.), the **indicator** of the structure if one exists (the preposition for a prepositional complement, the subordinator for a subordinate clause, etc.), and additional information if available (tense, number, etc.).

3.2 Pairing Syntactic Units

After finding all syntactic structures of interest, we traverse each structure to extract pairs that are connected by a syntactic relation (modifier, argument, subordinate clause). This means testing whether one structure is embedded in another, or whether they are at the same level, linked by a connective.

3.3 Semi-automatic Assignment of Semantic Relations to Pairs of Syntactic Units

Automatically Finding Suggestions for Semantic Relations. Our system starts with a minimum of manually encoded knowledge, and accumulates information as it processes texts. This design principle was adopted from TANKA [1]. The manually precoded knowledge consists of a dictionary of markers (subordinators, coordinators, prepositions). These markers are closed-class words, so not much effort is required to build such a resource. The system has the option to run without these resources, in which case it will take longer to begin making good predictions.

We apply memory-based learning, so that in every semantic relation assignment the system uses every previously processed example. This allows us to find the best match [15].

Every stored example is a tuple with the structure:

$$(word_{x1}, attr_{x1}, word_{x2}, attr_{x2}, relation)$$

where $word_{xi}$ is the head-word in structure x_i, and $attr_{xi}$ is a vector containing the structure's attributes listed in Section 3.1:

$$attr_x = (POS_{wordx}, SyntRole_x, Indic_x, OtherInfo)$$

Figure 1 presents the distance metric between two examples, represented as tuples.

The first option in our metric applies when a pair containing the same words as the current pair has already been tagged. The same two words may be connected by different semantic relations, if their attributes differ.
(1) *When* **you look** *at a cloud in the sky ...*
(2) **Look** *at the sky above* **you**.

In sentence (1) **you** is the subject, while in sentence (2) it is the prepositional complement. The two **(look,you)** pairs should be assigned different relations (AGENT in (1) and DIRECTION in (2)).

If we constrain the system to match only pairs of structures with the same attributes, generalization to pairs from different syntactic levels will not occur. The pairs **(protest,student)** from the sentences:
(3) *The students protested against tuition fee increase.*
(4) *student protest against tuition fee increase*
should both be assigned the AGENT relation, even though their attributes are obviously different.

$$P_i = [w_{i1}, a_{i1}, w_{i2}, a_{i2}, \text{Rel}]$$

$$dist(P_1, P_2) = \begin{cases} 0 & : \quad w_{11} = w_{21}, w_{12} = w_{22} \\ 0 & : min(d(net(w11), net(w21))) = 0 \\ d(P_1, P_2): & \quad otherwise \end{cases}$$

$net(w) = \{[w_1, w_2] | [w_1, w_2]$ extracted from sentence S, $w \in \{w_1, w_2\}\}$

$d(net(w_1), net(w_2)) = \sum_k d(P_{1k}, P_{2k}); P_{ik} \in net(w_i)$
$d([w_{11}, a_{11}, w_{12}, a_{12}, \text{Rel}], [w_{21}, a_{21}, w_{22}, a_{22}, \text{Rel}]) = \sum_k d(a_{12k}, a_{22k});$
(a_{ik} is an element in vector $attr_i$ associated withw_i)

$$d(a_x, a_y) = \begin{cases} 0: a_x = a_y; \\ 1: a_x \neq a_y \end{cases}$$

Fig. 1. Distance metric used for memory-based learning

The first option in the metric shows that we choose to allow the system to match tuples that do not have the same attributes, in order to let it generalize. The downside is that occasionally the metric will give inaccurate predictions.

When the words in two tuples P_1 and P_2 differ, we consider the distance between P_1 and P_2 to be 0 if the networks centered on the heads of P_1 and P_2 match. This idea was adopted from Delisle *et al.* [16] who applied it to verbs. We extend it to nouns.

A network centered on w consists of:

- a central vertex which represents w. It also contains syntactic and morphological information about w,
- a set of vertices connected with the central one, which represent the syntactic units from a sentence S and their syntactic attributes, with which w is connected through syntactic relation. w may be either the main element in relation with these units, or the modifier. For the sentence
  (5) *Weathermen watch the clouds day and night.*
  the system builds the following network centered on the verb:

```
[watch, v, svo,
 [weatherman,(sent,nil),(subj,nil),_],
 [cloud,(sent, nil),(compl, nil),_],
 [day_and_night,(sent,nil),(compl,nil),_]]
```

  The underscore replaces the semantic relation on that particular edge, which has not been yet assigned.

$d(net(w_1), net(w_2))$ shows how we compute the distance between two networks. It is the sum of distances between pairs of edges. Two edges match if the attributes of the word in vertices have the same syntactic role, and the same indicators. When we match edges, the actual words in the corresponding vertices do not matter, only their attributes. The best match will give the minimum distance. We only attempt to match networks centered on words with the same part of speech.

After processing each example, we will store the networks of tuples centered on both words in the example.

To show how the networks are matched, let us consider sentence (5) from above, and the network centered on the verb *watch*. The system will extract, from previously stored networks, those centered around verbs[1]. If sentence (6):
(6) *Air pilots know that clouds can bring rain, hail, sleet and snow.*
were processed before sentence (5), the system would find the following matching pattern:

```
[know, v, svo,
 [pilot,(sent,nil),(subj,nil),AGENT],
 [bring,(sent,nil),(compl,nil),OBJECT]]
```

The two networks match, because the centers of the networks match - the words have the same part of speech, and the same subcategorization pattern, and the edges match because the attributes of the words in the vertices match.

Because the edges with vertices **(watch,weatherman)** and **(know,pilot)** match, the AGENT relation for the pair **(know,pilot)** is proposed as a possible relation for **(watch,weatherman)**.

In the case where no matching network is found, the distance between two examples is computed as the distance between the modifiers. The key information is in particular the syntactic role and the indicator (if it exists). Our system works with a dictionary of indicators (prepositions, subordinators, coordinators), which are semantic relation markers. One indicator usually signals more than one relation (e.g. *since* may indicate a causal or a temporal relation).

After processing each sentence, the networks of pairs around head words are compiled and stored in memory for use with new examples.

4 Experiments

We need to compare our system with other knowledge acquisition systems available. There are no measures of time or precision that show how an automatic or user-based system performs.

The system that is most similar to ours is the one that we have started from, TANKA [11]. In order to compare the systems we will use the same input data – a text on meteorological phenomena [17] – the same syntactic analyser and the same evaluation measures. An exact comparison is not possible, since the two systems have different working paradigms. We will discuss this in Section 5.

After running the system in a set-up that allows us to compare it with TANKA, we run three more experiments, designed to evaluate its performance with a different list of semantic relations, and with a different parser.

[1] If more detailed information is available, the system will choose only networks associated with verbs that have the same subcategorisation structure (svo,svoc, etc.).

4.1 Parsers

We compare the performance of the system when it uses different syntactic analysers. We first use DIPETT [18], a comprehensive English parser. After using the results obtained to compare the system with TANKA, we plug in different parsers.

We have looked at the Link Grammar Parser [19] and the Xerox Incremental Parser (XIP) [20]. While the Link Grammar Parser is quite robust – it produces a parse tree for every input – its parse trees are too coarse-grained for the type of analysis that our system does. For example, for the sentence:

(7) These tiny clouds are real clouds

LINK produces the following output:

```
[S But [NP these tiny clouds NP] [VP are [NP real clouds NP] VP] . S]
```

We cannot extract modifier-noun relations from this parse tree.

XIP on the other hand produces relatively detailed parse trees. As a bonus for us, it also has the option to extract dependencies, which reduces our task of processing the parse tree looking for pairs. For the sentence

(8) Clouds tell the story.

the parser extracts the following information (apart from the parse tree):

DETD(story,the)
VDOMAIN(tell,tell)
VDOMAIN(cloud,cloud)
OBJ_POST(tell,story)
MAIN(cloud)
HEAD(story,the story)

Since XIP gives as a result pairs of syntactic units, we adjust the system to work with this output, without processing the parse tree.

The original TANKA system analysed three syntactic levels: clause, intra-clause and noun-phrase. For a better comparison with the new system, the list of syntactic units will have to contain structures from all these levels. The new system does not distinguish syntactic levels, but treats all structures the user wants uniformly. Table 1 shows the list of syntactic units that we ask the system to extract. These non-terminals come from DIPETT's code.

In Table 2 we show the list of non-terminals that describe the possible roles that each of the structures plays in a sentence.

XIP uses a much simpler grammar than DIPETT. We use the dependencies it detects to extract the data. The dependencies of interest are presented in Table 3.

The dependency relations also give us information about the syntactic roles that the words play in a sentence. They are shown in Table 4.

[2] The asterisk can be the empty string, or a string containing other dependency information, for example _PRE or _POST (it refers to the position of the modifier relative to the head), _PROGRESS (progressive verb), etc.

Table 1. List of syntactic units from DIPETT

adj	adjectives
n, proper_noun	nouns (common, proper)
advs, adv_clause, simple_adv_clause, pp_adv	adverbial modifiers (simple adverbs, adverbial clauses, adverbial phrases)
entity	covers anything that can be conceived of as an entity
predicate	head of a verb phrase
statement	clause
simple_sentence, complex_sentence	sentence (simple or compound)
subord_clauses, head_main_clause, next_main_clause	types of clauses (subordinate, main or coordinate)

Table 2. Possible syntactic roles in DIPETT

subj	subject
complement	complement
attrs	attributes
adverbial	adverbial
np_postmodifiers pre_modif post_modif	modifiers of the noun phrase
s_qualifier	sentence qualifier
rel_clause single_main_clause head_main_clause next_main_clause	type of clause
initial final medial	type of subordinate clause
ing_clause genitive_ing_clause to_infinitive_clause	type of relative clause

Table 3. List of dependency relations from XIP

MAIN	main element in the sentence
HEAD	head of a phrase
VDOMAIN*	head verb in a clause

Table 4. List of dependency relations that indicate syntactic roles from XIP

NMOD*	noun modifier
SUBJ*	subject
OBJ*	object
COMPL	complement
VMOD*	verb argument

4.2 Semantic Relations

The list of 47 semantic relations that we use combines three separate lists used in [1], one for each syntactic level that TANKA analysed. The semantic relations included are general, domain-independent. They are presented in [21].

Since the system is meant to be customisable, we experiment with plugging in a different list. This list contains 6 relations *causal, temporal, spatial, conjunctive, participant, quality.*

5 Results

The input text consisted of 513 sentences.

When DIPETT was plugged in, the experiment was performed by two judges (to make the assignment of semantic relations more objective) in 5 sessions of approximately 3 hours each. The overall time spent on semantic relation assignment was 6 hours, 42 minutes and 52 seconds. We have used the results collected from this run to automate the system when we changed the list of semantic relations, and when we changed the parser to XIP. Because the alternative list of

semantic relations we used is a generalised version of the original list, a simple mapping allowed us to change the results collected and the marker dictionary file.

Neither DIPETT nor XIP produced a correct parse for every sentence. When a complete parse (correct or incorrect) was not possible, DIPETT produced fragmentary parses. The semantic analyser extracted units even from tree fragments, although sometimes the fragments were too small to let us find pairs. XIP produces a parse tree for each input sentence, although not always a correct one.

Since XIP and DIPETT did not always parse correctly the same sentences, the pairs of concepts extracted by XIP were cross-referenced with the pairs tagged semi-automatically when DIPETT was plugged in, and then manually checked. Pairs obtained from XIP which were correctly identified were kept, even if the parse was erroneous (wrong part of speech, wrong phrase, etc.).

In the experiment with DIPETT, the semantic analyser extracted a total of 2020 pairs, 555 of which were discarded by the user in the dialogue step. An example of an erroneous pair comes from the sentence in example (9).

(9) *Tiny clouds drift across like feathers on parade.*

The semantic analyser produces the pair *(drift,parade)*, because of an erroneous parse tree, in which *parade* is parsed as a complement of *drift*, instead of a post-modifier for *feathers*. The correct pairing *(feather,parade)* will be missing, because it cannot be inferred from the parse tree.

XIP produced fewer correct parses than DIPETT. Its errors come mostly from mistagging words with part-of-speech information. For example, *clouds* is tagged mostly as a verb, even in structurally simple sentences. From the output produced, we extracted 1153 pairs, 445 of which were discarded.

Table 5 shows a summary of the results obtained, for the two parsers and the two lists of semantic relations (with 47 and 6 relations respectively), and the statistics of user actions (accept, choose, supply) during the semi-automatic memory-based semantic analysis step.

Table 5. Summary of results

Parser	nr. of rels	correct pairs	accept	choose	supply
DIPETT	47	1465	30.7% (450)	27.3% (401)	41.9% (614)
DIPETT	6	1465	49%% (718)	24.6% (360)	26.5% (388)
XIP	47	708	27.5% (195)	20.3% (144)	52.1% (369)
XIP	6	708	37% (262)	21.1% (150)	41.8% (296)

The results in Table 5 and the plots in Figures 2(1) and 2(2) show how the system behaves in 4 configurations: with two parsers (DIPETT and XIP), and two lists of relations (47 and 6 respectively). When the system works with the short list of relations it performs better for both parsers. Both lists make the system perform better with DIPETT than with XIP. This may be due to the amount of information that DIPETT provides, compared with XIP. Also, the system runs faster with DIPETT, when information about verb subcategorization allows it to filter out many networks before trying to match them. In

1. User action results for DIPETT, with 47 and 6 relations

2. User action results for XIP, with 47 and 6 relations

3. Comparison between case assignment in TANKA and our system

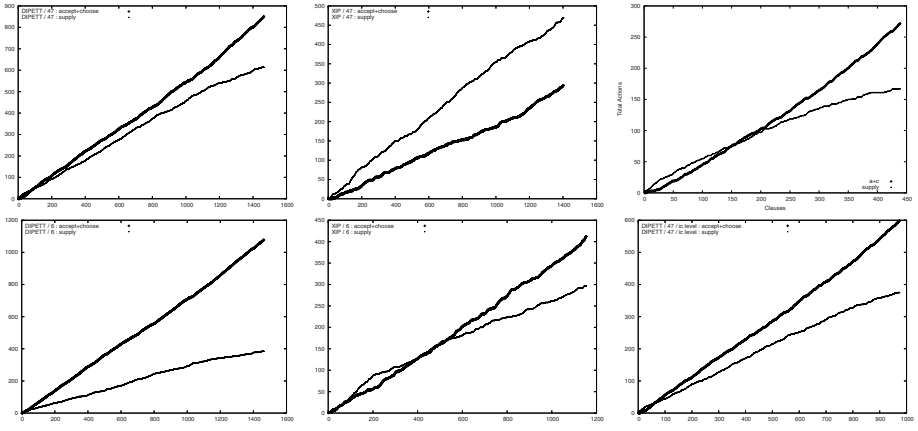

Fig. 2. Comparison of results with two parsers, and with TANKA

each figure the x axis shows the number of examples analysed, and the y axis shows the cummulative number of user actions (accept or choose versus supply). The plots show that as more examples are analysed, the system makes better suggestions.

For comparison of TANKA and our system, we present Figure 2(3). The first graph shows the user action results for the intra-clause level over the course of the experiment for the original TANKA system. Our system does not differentiate between syntactic levels, but based on the structures corresponding to each pair we can decide to which syntactic level it belongs. We have separated the results obtained for pairs from the intra-clause level, and present them for comparison in the second graph in Figure 2(3). The difference in the number of examples tagged comes from the fact that TANKA analyses the entire argument structure around the verb in one step, while our system tags each (argument,verb) pair separately.

We observe from these results that the new system starts learning much earlier. The original TANKA system processed half the input examples before the combined results of the *accept* and *choose* user actions surpassed *supply*; the new system obtains good results almost right away.

6 Evaluating the System's Customisability

The system's (Prolog) code is grouped in two modules: a module for extracting syntactic units and producing pairs, and a module for semantic analysis.

In order to allow the user to choose syntactic structures once a parser is plugged in, no modifications are necessary. During processing, the system automatically assigns a level label to the pair (*np* for noun-modifier pairs, *ic* for verb-arguments pairs, *cl* for pairs of clauses). The user can just set a parameter to *np, ic* or *cl* to choose the level she is interested in. For a more fine-grained selection, the user can access a detailed list of non-terminals used by the parser. When we do not have access to the parser's grammar, a list of nonterminals can be extracted from the parse trees produced. A tool that performs this task is part of future work.

Plugging in a new syntactic parser has various degrees of difficulty. If the structure of the output it produces matches the one obtained with DIPETT, no change is required in the code. Otherwise, the system must be provided with a description of the grammar for the new parser. In the case of XIP, the parser itself produces a list of dependencies, so the system was adjusted to bypass the tree processing stage, and its rules for finding syntactic roles and extract indicators were modified.

Plugging in a different list of semantic relations requires modifying one rule in the semantic analysis module (the rule simply lists the possible semantic relations to be assigned) and, optionally, modifying the dictionary containing 325 markers. While the system will function without this dictionary, its performance will drop since it needs either indicators or previously tagged examples to find semantic relations. In our experiments, we have used a list of 6 relations that generalize the original list of 47, so the dictionary change was automatic; we manually built a hash table to indicate the mapping between the two lists.

7 Conclusions

Having a human judge supervise the task of semantic analysis produces accurate results, but the time needed to spend on the task may be prohibitively long. Also, the type of knowledge that one wants to extract from a text, and the semantic relations to assign to it may vary. We propose a semi-automatic semantic analysis system, customisable to the task at hand. It can use different syntactic analysers, it will extract the syntactic units that the user is interested in, and will tag them with the semantic labels that are relevant to the domain of the input text.

We have compared our system with a similar endeavour. The results show that having a unified approach to analysing text leads to better results, in the form of faster learning. The learning that the system performs is memory-based, in which all examples previously analysed are used when processing a new one.

Part of future work is to deploy the system on the Web, so that it can be used for semantic analysis with various configurations. We also aim to refine and improve our system's learning part by using machine learning tools and lexical resources. We experimented with using other methods other than memory based learning, and lexical resources such as WordNet and Roget's Thesaurus. The experiments performed with base noun-phrases were promising [21], and we plan to incorporate these resources in our system.

References

1. Barker, K., Delisle, S., Szpakowicz, S.: Test-driving TANKA: Evaluating a semi-automatic system of text analysis for knowledge acquisition. In: Canadian AI, Vancouver, BC, Canada (1997) 60–71
2. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The Berkeley FrameNet project. In: COLING-ACL, Montreal, Canada (1998) 86–90
3. Rosario, B., Hearst, M.: Classifying the semantic relations in noun-compounds via a domain specific hierarchy. In: EMNLP, Pittsburg, PA, USA (2001) 82–90
4. Gomez, F.: A representation of complex events and processes for the acquisition of knowledge from text. Kowledge-Based Systems **10** (1998) 237–251
5. Fillmore, C., Atkins, B.T.: FrameNet and lexicographic relevance. In: LREC, Granada, Spain (1998)
6. Gildea, D., Jurafsky, D.: Automatic labeling of semantic roles. Computational Linguistics **28** (2002) 245–288
7. Hull, R.D., Gomez, F.: Semantic interpretation of nominalizations. In: 13th National Conference on Artificial Intelligence, Portland, Oregon, USA (1996) 1062–1068
8. Rosario, B., Hearst, M., Fillmore, C.: The descent of hierarchy and selection in relational semantics. In: ACL, Philadelphia, PA, USA (2002)
9. Clark, P., Porter, B.: Building concept reprezentations from reusable components. In: AAAI, Providence, Rhode Island (1997) 367–376
10. Fan, J., Barker, K., Porter, B., Clark, P.: Representing roles and purpose. In: KCAP. (2001) 38–43
11. Barker, K.: Semi-Automatic Recognition of Semantic Relationships in English Technical Texts. PhD thesis, University of Ottawa, Department of Computer Science (1998) http://www.cs.utexas.edu/users/kbarker/thesis.
12. Kilgarriff, A., Tugwell, D.: WORD SKETCH: Extraction and display of significant collocations for lexicography. In: Workshop on Collocation: Computational Extraction, Analysis and Exploitation, 39th ACL & 10th EACL, Toulouse, France (2001) 32–38
13. Lin, D., Pantel, P.: Concept discovery from text. In: COLING, Taipei, Taiwan (2002) 577–583
14. Pantel, P., Lin, D.: Discovering word senses from text. In: SIGKDD, Edmonton, Canada (2002) 613–619
15. Daelemans, W., van den Bosch, A., Zavrel, J.: Forgetting exceptions is harmful in language learning. Machine Learning **34** (1999) 11–34
16. Delisle, S., Copeck, T., Szpakowicz, S., Barker, K.: Pattern matching for case analysis: A computational definition of closeness. In: ICCI, Sudbury, ON, Canada (1993) 310–315
17. Larrick, N.: Junior Science Book of Rain, Hail, Sleet and Snow. Garrard Publishing Company, Champaign, Illinois (1961)
18. Delisle, S., Szpakowicz, S.: Realistic parsing: Practical solutions of difficult problems. In: PACLING, Brisbane, Queensland, Australia (1995)
19. Temperley, D., Sleator, D., Lafferty, J.: The LINK parser (1998) http://www.link.cs.cmu.edu/link.
20. Chanod, J.P., Ait-Mokhtar, S., Roux, C.: Xerox Incremental Parser, ongoing research (2004) Xerox Research Centre Europe.
21. Nastase, V., Szpakowicz, S.: Exploring noun-modifier semantic relations. In: International Workshop on Computational Semantics, Tillburg, Netherlans (2003)