Customisable Semantic Analysis of Texts

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 semantic relations to be used. The system examines parse trees to decide if there is interaction between concepts that underlie syntactic units. Memory-based learning tags new unit pairs with semantic labels from a user-defined set.

1 Introduction

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 include 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.

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

The system extracts pairs of concepts from the text. 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 that no set of semantic relations is appropriate for all NLP tasks. Semantic relations are assigned semi-automatically. The user can accept a unique suggestion made by the system, choose from a list, enter the correct answer manually or reject the pair.

Barker et al. (1997) 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), to recognize 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.

2 Related Work

One style of semantic analysis for knowledge acquisition uses predefined templates, filled with information from processed texts (Baker et al., 1998). In other systems lexical resources are specifically tailored to meet the requirements of the domain (Rosario and Hearst, 2001) or of the system (Gomez, 1998). Such systems extract information from some types of syntactic units (clauses – (Fillmore and Atkins, 1998), (Gildea and Jurafsky, 2002), (Hull and Gomez, 1996); noun-phrases – (Hull and Gomez, 1996), (Rosario et al., 2002)). 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 (Clark and Porter, 1997). 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) (Fan et al., 2001). The relations come from three syntactic levels (Barker, 1998).

In purely statistical approaches that traverse corpora to establish connections between concepts based on word collocations, the incidence of errors is not negligible (Kilgarriff and Tugwell, 2001), (Lin and Pantel, 2002), (Pantel and Lin, 2002).

In our system, user feedback will help 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 (Rosario and Hearst, 2001). In others it is important to see how entities are related to an event, e.g., in legal texts (Baker et al., 1998). 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 with good coverage and detailed syntactic information. The parse trees give us syntactic units, from which we choose those of interest to the user. To pair units 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. It must conform to 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 on the same level, linked by a connective.

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

3.3.1 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 idea was adopted from TANKA (Barker et al., 1997). The manually encoded 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 start making good predictions.

We apply memory-based learning, so that in every semantic label assignment the system uses every previously processed example. This allows us to find the best match (Daelemans et al., 1999).

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 described in Section 3.1:

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

Figure 1 presents the distance metric between two pairs of words.

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 a subject, while in sentence (2) it is a 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.

We choose to allow the system to match pairs 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 pairs P_1 and P_2 differ, we consider the distance between P_1 and P_2 to be 0 if the networks of pairs centered on the heads of P_1 and P_2 match. This idea was adopted from Delisle et al. (1993) who applied it to verbs. We extend it to nouns.

A network of pairs centered on w consists of the collection of pairs from a sentence S, in which w appears either as the main (head) element, either as an argument (modifier). When we compute the distance between two networks, we compute the distance between pairs in the two networks. Two pairs match if their modifiers have the same syntactic role, and the same indicators. The best match will give the minimum distance. We only attempt to match networks centered on words with the same part of speech. For the sentence:

(5) *Weathermen watch the clouds day and night.*,

the system builds the following network centered on the predicate *watch*:

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

The system will extract, from previously stored networks, those centered around verbs[1] For example, 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,
 [_X,(sent,nil),(subj,nil),AGENT],
 [_Y,(sent,nil),(compl,nil),OBJECT]]
```

According to the metric, the networks match, and the pairs **(watch,weatherman)**

[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.).

$P_i = [w_{i1}, w_{i2}]$

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

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

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

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

Figure 1: Distance metric used for memory-based learning

and **(know,_X)** match, so the relation for pair **(know,_X)** is proposed as a possible relation for pair **(watch,weatherman)** .

If no matching network has been found, the distance between two pairs 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 may signal 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.

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 (Barker, 1998). In order to compare the systems we will use the same input data – a text on meteorological phenomena (Larrick, 1961) – 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.

4.1 Parsers

We compare the performance of the system when it uses different syntactic analysers. We first use DIPETT (Delisle and Szpakowicz, 1995), 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 (Temperley et al., 1998) and the Xerox Incremental Parser (XIP) (Chanod et al., 2004). 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)

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.

`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 complex)
`subord_clauses, head_main_clause, next_main_clause`	types of clauses (subordinate, main or coordinate)

Table 1: List of syntactic units from DIPETT

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 here 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.

4.2 Semantic Relations

The list of 47 semantic relations that we use combines three separate lists used in (Barker et al., 1997), one for each syntactic level that TANKA analysed. The semantic relations included are general, domain-independent.

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*.

[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 modifi er relative to the head), _PROGRESS (progressive verb), etc.

`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 2: Possible syntactic roles in DIPETT

`MAIN`	main element in the sentence
`HEAD`	head of a phrase
`VDOMAIN*`[2]	head verb in a clause

Table 3: List of dependency relations from XIP

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 all trees. When a complete parse (cor-

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

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

rect 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.

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, even if the parse was erroneous (wrong part of speech, wrong phrase, etc.), were kept.

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 number of each possible user action (accept, choose, supply) during the memory-based semantic analysis step.

The results in Table 5 and the plots in Figures 2 and 3 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 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.

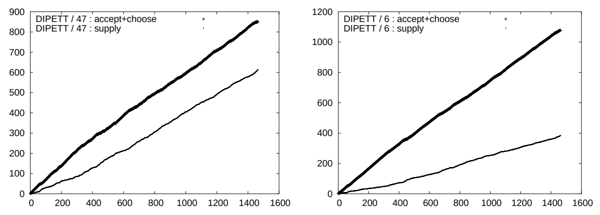

Figure 2: User action results for DIPETT

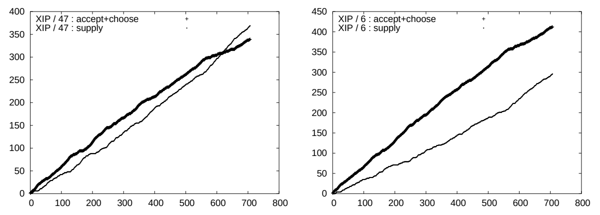

Figure 3: User action results for XIP

For comparison of the TANKA and our system, we present Figures 4 and 5. Figure 4 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 which syntactic level it pertains to. We have separated the results obtained for pairs from the

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)

Table 5: Summary of results

intra-clause level, and present them for comparison in Figure 5. 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.

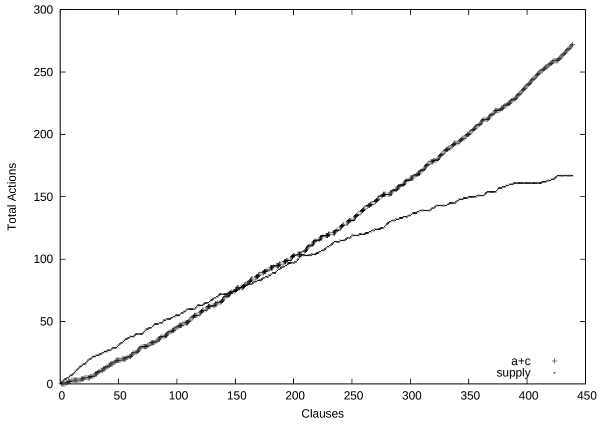

Figure 4: Original TANKA: User action over time for intra-clause level

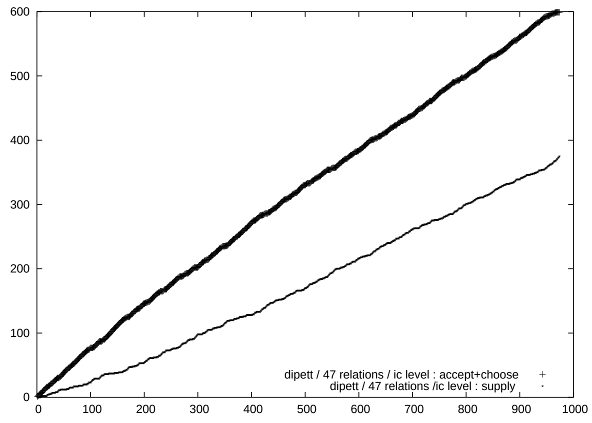

Figure 5: New system: Statistics for the intra-clause level

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 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 a 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 labels 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 for 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. Also, its learning part could be refined using machine learning tools and lexical resources.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *COLING-ACL*, pages 86–90, Montreal, Canada.

Ken Barker, Sylvain Delisle, and Stan Szpakowicz. 1997. Test-driving TANKA: Evaluating a semi-automatic system of text analysis for knowledge acquisition. In *Canadian AI*, pages 60–71, Vancouver, BC, Canada.

Ken Barker. 1998. *Semi-Automatic Recognition of Semantic Relationships in English Technical Texts*. Ph.D. thesis, University of Ottawa, Department of Computer Science. http://www.cs.utexas.edu/users/kbarker/thesis.

Jean-Pierre Chanod, Salah Ait-Mokhtar, and Claude Roux. 2004. Xerox Incremental Parser, ongoing research. Xerox Research Centre Europe.

Peter Clark and Bruce Porter. 1997. Building concept reprezentations from reusable components. In *AAAI*, pages 367–376, Providence, Rhode Island.

Walter Daelemans, Antal van den Bosch, and Jakub Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34(1):11–34.

Sylvain Delisle and Stan Szpakowicz. 1995. Realistic parsing: Practical solutions of difficult problems. In *PACLING*, Brisbane, Queensland, Australia.

Sylvain Delisle, Terry Copeck, Stan Szpakowicz, and Ken Barker. 1993. Pattern matching for case analysis: A computational definition of closeness. In *ICCI*, pages 310–315, Sudbury, ON, Canada.

James Fan, Ken Barker, Bruce Porter, and Peter Clark. 2001. Representing roles and purpose. In *KCAP*, pages 38–43.

Charles Fillmore and Beryl T. Atkins. 1998. FrameNet and lexicographic relevance. In *LREC*, Granada, Spain.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Fernando Gomez. 1998. A representation of complex events and processes for the acquisition of knowledge from text. *Kowledge-Based Systems*, 10(4):237–251.

Richard D. Hull and Fernando Gomez. 1996. Semantic interpretation of nominalizations. In *13th National Conference on Artificial Intelligence*, pages 1062–1068, Portland, Oregon, USA.

Adam Kilgarriff and David Tugwell. 2001. WORD SKETCH: Extraction and display of significant collocations for lexicography. In *Workshop on Collocation: Computational Extraction, Analysis and Exploitation, 39th ACL & 10th EACL*, pages 32–38, Toulouse, France.

Nancy Larrick. 1961. *Junior Science Book of Rain, Hail, Sleet and Snow*. Garrard Publishing Company, Champaign, Illinois.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *COLING*, pages 577–583, Taipei, Taiwan.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *SIGKDD*, pages 613–619, Edmonton, Canada.

Barbara Rosario and Marti Hearst. 2001. Classifying the semantic relations in noun-compounds via a domain specific hierarchy. In *EMNLP*, pages 82–90, Pittsburg, PA, USA.

Barbara Rosario, Marti Hearst, and Charles Fillmore. 2002. The descent of hierarchy and selection in relational semantics. In *ACL*, Philadelphia, PA, USA.

Davy Temperley, Daniel Sleator, and John Lafferty. 1998. The LINK parser. http://www.link.cs.cmu.edu/link.