# Polaris: Lymba's Semantic Parser

**Dan Moldovan** and **Eduardo Blanco**

Lymba Corporation
Richardson, TX 75080 USA
{moldovan,eduardo}@lymba.com

**Abstract**

Semantic representation of text is key to text understanding and reasoning. In this paper, we present Polaris, Lymba's semantic parser. Polaris is a supervised semantic parser that given text extracts semantic relations. It extracts relations from a wide variety of lexico-syntactic patterns, including verb-argument structures, noun compounds and others. The output can be provided in several formats: XML, RDF triples, logic forms or plain text, facilitating interoperability with other tools. Polaris is implemented using eight separate modules. Each module is explained and a detailed example of processing using a sample sentence is provided. Overall results using a benchmark are discussed. Per module performance, including errors made and pruned by each module are also analyzed.

**Keywords:** Semantic relations, semantic parser, semantic representation of text

## 1. Introduction

Extracting semantic relations from text is an important step towards understanding the meaning of text. Broadly speaking, a semantic relation is a directional connection between two concepts. For example, in the sentence *John and Mary bought a brand new BMW convertible last year*, *John* and *Mary* are the AGENT of *bought*, *convertible* the THEME and *last year* the TIME of *bought*. In addition to these verb-argument relations, there are more relations: *BMW* MAKES *convertibles* and the *convertible* has VALUE *brand new*.

Semantic parsers are tools that extract meaning from text. The means to do so include semantic relations like in the above example, extensions to first order logic (Poon and Domingos, 2009), logical forms (Allen et al., 2008), and other formal representations (Bos, 2008). Semantic relation extraction has received considerable attention, including numerous competitions with dozens of participants (Carreras and Màrquez, 2005; Diab et al., 2007; Màrquez et al., 2007; Girju et al., 2007; Ruppenhofer et al., 2009; Hendrickx et al., 2009; Pustejovsky and Verhagen, 2009).

In this paper we present Polaris, a semantic parser that takes free English text or parsed sentences and extracts a rich set of semantic relations. Polaris extracts semantic relations from a wide variety of lexico-syntactic patterns (Section 3), not only verb-argument structures like semantic role labelers do. Polaris will be freely available for research purposes[1] and can provide its output in several formats: XML, RDF triples, logic forms or plain text. Polaris is also commercially available and it is at the core of many software products developed at Lymba: question answering system (Moldovan et al., 2010), entailment recognition (Tatu and Moldovan, 2005), ontology creation (Balakrishna et al., 2010), etc.

Whereas syntactic parsers have matured during the last decade, the progress in semantic parsers has been more modest mainly due to the complexity of the problem. Role labelers extract only relations between a verb and its arguments, however SemEval tasks have focused on relations between nominals. Working with semantics is harder than some other NLP tasks mainly because: (1) there is no agreement on the set of relations to extract; (2) relations are often poorly defined (a sentence and a couple of examples); and (3) relations can be encoded between a wide variety of arguments and syntactic patterns.

Polaris extracts semantic relations that are easily transformed into RDF triples, a standard for knowledge representation proposed by W3C and recommended for knowledge interchange within the semantic web. Its output can be readily used with third party RDF management tools such as Oracle 11g and RDF reasoners such as AllegroGraph.

## 2. Previous Work

There have been several proposals to extract semantic relations from text. The public availability of large corpora (e.g., PropBank, FrameNet) allows to reliably train supervised models. Supervised approaches typically focus on a fixed set of semantic relations (AGENT, MANNER, etc.) and consider relations between arguments fulfilling some constraints, e.g., forming a noun compound (Tratz and Hovy, 2010), being a verb and one of its syntactic arguments (Gildea and Jurafsky, 2002), being an instance of a pattern that usually encodes a particular relation (Girju, 2003).

Polaris is a comprehensive effort to extract relations from text, blending previous efforts on relation extraction and incorporating in-house annotations. Polaris is an unified, self-contained and ready-to-use tool and it extracts more relations than any other single tool available (Figure 1).

Our approach to representing text semantics contrasts with first order logic and with work grounded on extensions of first order logic (Poon and Domingos, 2009). We believe that using a fixed set of dyadic relations is better suited for automated reasoning than allowing an uncontrollable large number of predicates with variable number of arguments.

A novelty of Polaris, that brings a significant improvement, is its feature of imposing semantic restrictions on relations arguments which in turn results in filtering out relations that cannot exist between certain arguments. It is grounded on an extended definition for semantic relations that specifies semantic restrictions on the relation arguments.

[1] http://www.lymba.com

The terrorists sent letter bombs a few years ago to newspaper offices in New York City and Washington, D.C., [...]

AGENT(*sent*, *The terrorists*) THEME(*sent*, *letter bombs*) IS-A(*letter*, *letter bomb*) IS-A(*bomb*, *letter bomb*) PURPOSE(*letter bombs*, *bombs*) TIME(*sent*, *a few years ago*) LOCATION(*sent*, *to newspaper [...] D.C.*) MAKE(*offices*, *newspaper*) RECIPIENT(*letter bombs*, *newspaper [...] D.C.*) IS-A(*offices*, *newspaper offices*) LOCATION(*newspaper offices*, *New York City*) LOCATION(*newspaper offices*, *Washington*) PART-WHOLE(*D.C.*, *Washington*) LOCATION(*Washington*, *D.C.*)

Figure 1: Example of thorough semantic representation Polaris extracts. Only four out of fifteen relations are semantic roles, i.e., a semantic relation whose arguments correspond to a verb and one of its arguments.

| | Relation | Example | DOMAIN × RANGE |
|---|---|---|---|
| CAU | CAUSE | CAU(*tsunami*, *earthquake*) | [si] × [si] |
| JST | JUSTIFICATION | JST(*don't smoke*, *it is forbidden*) | [si] × [si ∪ ntao] |
| IFL | INFLUENCE | IFL(*poor grade*, *missing classes*) | [si] × [si] |
| INT | INTENT | INT(*Mary*, *buy*) | [aco] × [si] |
| PRP | PURPOSE | PRP(*garage*, *storage*) | [si ∪ co ∪ ntao] × [si ∪ ntao] |
| VAL | VALUE | VAL(*car*, *blue*) | [o ∪ si] × [ql] |
| SRC | SOURCE | SRC(*avocados*, *Mexican*) | [o] × [loc ∪ ql ∪ ntao ∪ ico] |
| AGT | AGENT | AGT(*gave*, *John*) | [si] × [aco] |
| EXP | EXPERIENCER | EXP(*felt*, *John*) | [si] × [o] |
| INS | INSTRUMENT | INS(*broke*, *a hammer*) | [si] × [co ∪ ntao] |
| THM | THEME | THM(*gave*, *flowers*) | [ev] × [o] |
| TPC | TOPIC | TPC(*discuss*, *issue*) | [ev] × [o ∪ si] |
| STI | STIMULUS | STI(*listen*, *train*) | [ev] × [o] |
| ASO | ASSOCIATION | ASO(*phone*, *fax*) | [ent] × [ent] |
| KIN | KINSHIP | KIN(*John*, *his cousin*) | [aco] × [aco] |
| ISA | IS-A | ISA(*car*, *convertible*) | [o] × [o] |
| PW | PART-WHOLE | PW(*car*, *engine*) | [o] × [o] ∪ [l] × [l] ∪ [t] × [t] |
| MAK | MAKE | MAK(*BMW*, *cars*) | [co ∪ ntao] × [co ∪ ntao]] |
| POS | POSSESSION | POS(*John*, *truck*) | [co] × [co] |
| MNR | MANNER | MNR(*delivery*, *quick*) | [si] × [ql ∪ st ∪ ntao] |
| RCP | RECIPIENT | RCP(*gave*, *Mary*) | [ev] × [co] |
| SYN | SYNONYMY | SYN(*twelve*, *a dozen*) | [ent] × [ent] |
| LOC | LOCATION | LOC(*gave*, *in the porch*) | [o ∪ si] × [loc] |
| TMP | TIME | TMP(*gave*, *yesterday*) | [o ∪ si] × [tmp] |
| PRO | PROPERTY | PRO(*John*, *height*) | [o ∪ si] × [ntao] |
| QNT | QUANTIFICATION | QNT(*roses*, *a dozen*) | [o ∪ si] × [qn] |

Table 1: Relation inventory used by Polaris. Domain and range restrictions are defined using the ontology presented in Section 3.1.

## 3. Approach

Polaris aims at extracting semantic relations from a wide variety of lexico-syntactic patterns, including verb argument structures (e.g., *John runs fast*: AGENT(*runs*, *John*), MANNER(*runs*, *fast*)), nominals (e.g., *door knob*: PART-WHOLE(*door*, *door knob*)), genitives (e.g., *Mary's house*: POSSESSION(*Mary*, *house*)), adjectival phrases (e.g., *cat in the tree*: LOCATION(*cat*, *the tree*)), adjectival clauses (e.g., *the man who killed Kennedy*: AGENT(*killed*, *the man*)) and others. Figure 1 shows a sample sentence and the semantic relations found in it. We denote a semantic relation R holding between $x$ and $y$ R($x$, $y$). R($x$, $y$) means $x$ *has* R $y$, e.g., AGENT(*bought*, *John*) means *bought has* AGENT *John*.

The relation set (Table 1) is fixed and was decided based on inventories used by others (PropBank, FrameNet, NomBank, SemEval competitions) and our own annotations. Some relations considered elsewhere are ignored since they do not occur frequently enough in text and their automatic extraction would not be feasible, e.g., ENTAILMENT.

Polaris uses an extended definition for semantic relations. Whereas most relation inventories are defined using plain English and some examples, Polaris incorporates semantic restrictions on domains and ranges (i.e., what kind of concepts can be the first and second argument). These restrictions are defined in Table 1 using a modified version (Section 3.1) of an ontology first proposed by Helbig (2005). Domain and range restrictions allow us to select plausible relations that may hold between any pair of concepts ($x$, $y$) simply by enforcing that their semantic classes are compatible with the relation definition. For example, DOMAIN(INTENT) is restricted to animate concrete objects, thus, INTENT is not proposed for the argument pair (*wind*, $y$) since an abstract object like *wind* cannot be the first argument of INTENT by definition.

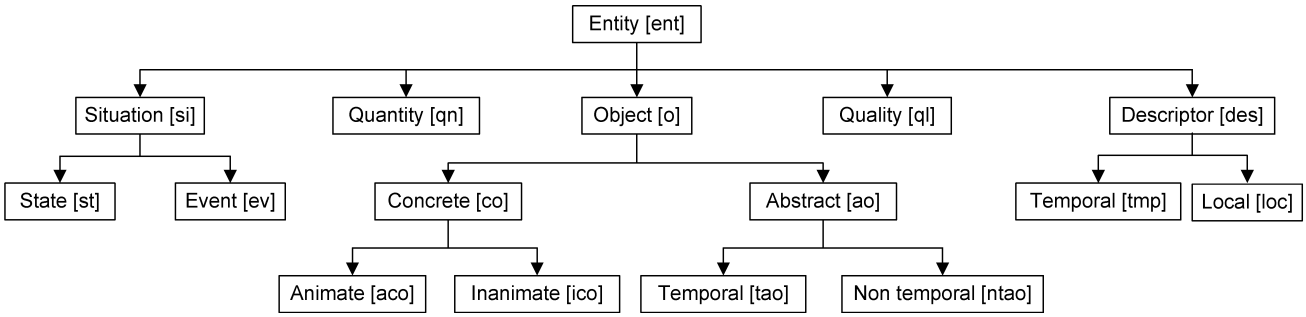


Figure 2: The ontology of sorts and their acronyms.

### 3.1. Ontology of Sorts

In order to define domains and ranges for each relation, we use a customized ontology of sorts (Figure 2) modified from the one proposed by Helbig (2005). The root corresponds to entities, which refers to *all things about which something can be said*.

Objects can be either concrete or abstract. The former occupy space, are touchable and tangible. The latter are intangible; they are somehow a product of human reasoning. Concrete objects are further divided into animate or inanimate. The former have life, vigor or spirit (e.g., *John*); the latter are dull, without life (e.g., *table*, *pencil*). Abstract objects are divided into temporal or non temporal. The first corresponds to abstractions regarding points or periods of time (e.g., *July*, *last week*); the second to any other abstraction (e.g., *disease*, *justice*). Some abstract objects can be sensually perceived, e.g., *pain*, *odor*.

Situations are anything that happens at a time and place. Simply put, if one can think of the time and location of an entity, it is a situation. Events (e.g., *go*, *grow*) imply a change in the status of other entities, states (e.g., *standing next to the door*, *account for 10% of sales*) do not. Situations can be expressed by verbs (e.g., *move*, *print*) or nouns (e.g., *party*, *hurricane*).

Descriptors complement entities by stating properties about their spatial or temporal context. They are composed of an optional non-content word signaling the local or temporal context and another entity. Local descriptors are further composed of a concrete object or situation, e.g., $[above]_{prep}$ $[the\ roof]_{co}$; temporal descriptors by a temporal abstract object or situation, e.g., $[during]_{prep}$ $[the\ party]_{ev}$. The non-content word signaling the local or temporal context is usually present, but not always, e.g., *The* $[birthplace]_{ev}$ *of his mother is* $[Ankara]_{loc}$.

Qualities represent characteristics that can be assigned to entities. They can be quantifiable like *tall* and *heavy*, or unquantifiable like *difficult* and *sleepy*. Quantities represent quantitative characteristics of concepts, e.g., *a few pounds*.

## 4. Architecture

Polaris is implemented using eight modules. These modules form a pipeline and currently there are no feedbacks: each module only feeds the next one.

1. **Pre-processing.** Tokenizer, POS tagger, named entity recognizer, syntactic parser, word sense disambiguator and co-reference resolution.
2. **Bracketer.** Parse tree is slightly modified to facilitate argument detection.
3. **Argument Detection.** Argument pairs likely to encode a semantic relation are selected and assigned candidate relations based on lexico-syntactic patterns.
4. **Domain and Range Filtering.** Candidates assigned to each argument pair are filtered based on the arguments' semantic classes and the extended definition of each relation.
5. **Grouping.** Argument pairs are clustered into nine generic syntactic patterns: `V_V`, `V_N`, `V_J`, `N_V`, `N_N`, `N_J`, `J_V`, `J_N`, `J_J`.
6. **Feature Extraction.** Different feature sets are extracted depending on the generic pattern.
7. **Classifiers.** SVM, Semantic Scattering (Moldovan and Badulescu, 2005), Decision Trees and Naive Bayes are used, both in a per-relation and per-pattern approach. After combining the output of the classifiers, each candidate of each argument pair is assigned a confidence score.
8. **Conflict Resolution.** Classifiers' output is resolved and tuned to avoid conflicts. Final semantic representation is obtained.

Module 1 can be bypassed if the annotations have already been extracted. The implementation allows the user to easily deactivate a module, retrain using only specific relations (commonly, the less relations the more accurate), focus on certain syntactic patterns and incorporate new annotations. Below we give more details for each module. Figure 3 exemplifies the output of each module for the input sentence *The two security guards are carrying automatic weapons*.

Pre-processing is done with in-house tools that obtain state-of-the-art performance. The bracketer is specially useful to detect relations within a noun phrase. In Figure 3, the parser creates a base NP (an NP without embedded NPs) for *the two security guards*. After bracketing, two new NPs are created (i.e., $[the\ [two\ [security\ guards]_{NP}]_{NP}]_{NP}$), facilitating the extraction of argument pairs *(security guards, two)* and *(security guards, guards)*, since all arguments now correspond to a single node in the parse tree.

Argument Detection is based on the lexico-syntactic patterns seen during training. This step is key: in order to

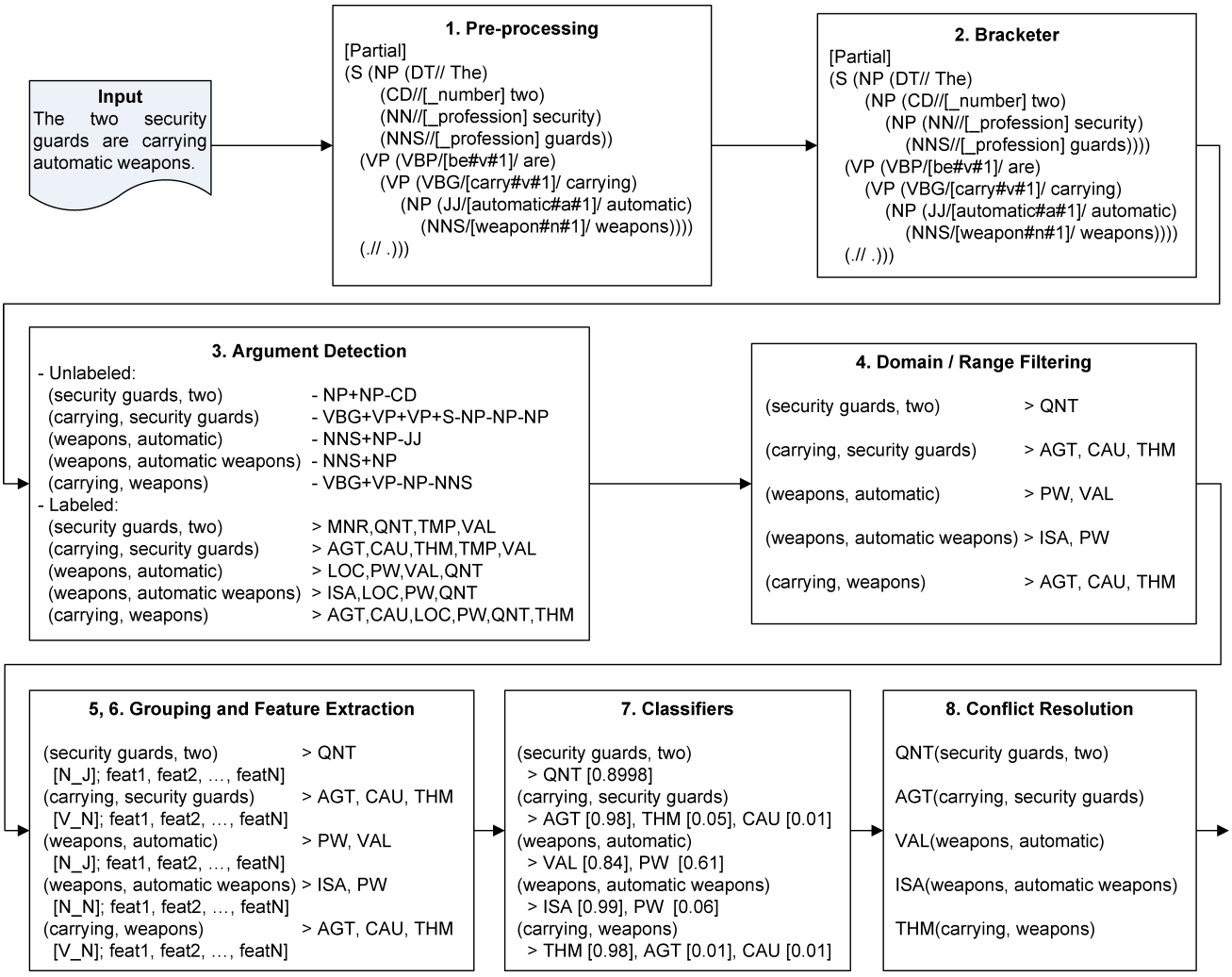


Figure 3: Polaris modules and their output for the sample sentence *The two security guards are carrying automatic weapons.*

extract a relation R holding between two concepts, we must first detect that the concepts are likely to encode any semantic relation. This module uses two steps: unlabeled and labeled argument detection. Unlabeled argument detection extracts argument pairs likely to encode a relation, and labeled argument detection assigns candidate relations to the extracted argument pairs. Both tasks are done based on counts over the training data.

Domain and range filtering is done in two phases. First, argument pairs are assigned a semantic class from the ontology. Second, the candidate relations whose domain and range are not compatible with the semantic classes of the argument pair are discarded as candidates.

Assigning a semantic class from the ontology to an arbitrary piece of text is not a trivial task. First, the head word of a potential argument is identified. Then, the head is mapped into a semantic class from the ontology using three sources of information: POS tags, WordNet hypernyms and named entity (NE) types. We obtained rules that define the mapping following a data-driven approach using a subset of the available data. For this task, we do not use word senses because in our experiments it did not bring any improvement; all senses are considered for each word. Rules are of the following form:

```
aco = neType=(human|organization|country
              |town|province|other-loc)
  || (POStag=noun &&
      ((isHypo(entity.n.1) &&
        !isHypo(thing.n.9,anticipation.n.4))
       || isHypo(social_group.n.1)))
```

Note that the above rule partially accounts for metonymy resolution. NE types like `organization` and `location` are mapped to animate concrete object even though they can be so only when metonymy is used, e.g., *[The White house]*$_{\text{organization}}$ *passed an important bill*, *[Washington]*$_{\text{location}}$ *passed an important bill*.

The next step is to group concept pairs into simplified syntactic paths. These groups are defined by the category of the argument head, `V` stands for verb, `N` for noun and `J` for adjective. This allow us to extract features and create models specialized on each generic pattern.

Feature extraction considers standard features for semantic relation extraction (syntactic path, first word, voice, etc), as well as features that we have identified over time. The latter include, among many others, the semantic class of modifier noun, useful to detect relations within genitives (Moldovan and Badulescu, 2005).

For each candidate of each argument pair, the classifiers assign a confidence score ranging from 0 to 1. Finally, con-

| Disabled Module(s) | P | R | F |
|---|---|---|---|
| 2: Bracketer | 0.274 | 0.723 | 0.398 |
| 4: Domain and Range Filtering | 0.230 | 0.710 | 0.354 |
| 8: Conflict Resolution | 0.231 | 0.761 | 0.353 |
| 2 and 4 | 0.202 | 0.716 | 0.315 |
| 2 and 8 | 0.203 | 0.765 | 0.321 |
| 4 and 8 | 0.203 | 0.729 | 0.317 |
| 2, 4 and 8 | 0.182 | 0.745 | 0.293 |
| None (all enabled) | 0.321 | 0.731 | 0.446 |

Table 2: Overall performance disabling different combinations of modules.

flict resolution combines the output of the classifiers and creates the final semantic representation. Semantic relations with low confidence score are discaded and conflicts resolved. In the example in Figure 3, conflict resolution picks the candidate with highest confidence score for each argument pair and no relations are discarded.

## 5. Results and Performance Analysis

Polaris uses as training corpora a mixture of publicly available resources (FrameNet, PropBank NomBank and SemEval competitions; their relations inventories were mapped to our inventory) and annotations done at Lymba over time. The latter include questions from several TREC competitions and text sources relevant to past projects at Lymba. Features are extracted manipulating the output of automatic tools, i.e., features in training potentially contain the same kind of errors found when using Polaris in a real application.

For testing purposes, we have fully annotated a benchmark that contains text outside the domain of the training data. This benchmark has been annotated without any sort of restriction on the kind of arguments that may encode a relation: any relation (from our inventory) holding between any two concepts is considered. This is a significant step towards realistic evaluation of semantic parsers: the parser's output is tested against a gold benchmark containing relations between concepts that were simply ignored during training. As a result, unlike evaluations of other tools for relation extraction, we evaluate Polaris against a much tougher benchmark.

### 5.1. Overall results

Table 2 provides overall results disabling different combinations of modules. The best results are obtained when all modules are enabled and the worst when the bracketer, domain and range filtering, and conflict resolution are disabled. It is worth noting that only disabling domain and range filtering brings a decrease in f-measure of $0.092$, and disabling conflict resolution a decrease of $0.093$. On the other hand, disabling the bracketer brings a more moderate decrease of $0.048$. Disabling either the bracketer or domain and range filtering brings small losses in recall ($0.008$ and $0.021$), but bigger losses in precision ($0.047$ and $0.091$ respectively). In contrast, disabling conflict resolution implies an increase of $0.030$ in recall, but a decrease of $0.090$ in precision, from $0.321$ to $0.231$. This is due to the nature of the conflict resolution module: out of all relations predicted by the classifiers, it discards the ones that are incompatible with each other. Enabling conflict resolution successfully increases precision at the cost of a small penalty in recall (i.e., most relations discarded are not valid, but a few are wrongly discarded).

| **Unlabeled argument detection** | |
|---|---|
| Correct | #argpairs extracted corresponding to a relation |
| Spurious | #argpairs extracted not corresponding to a relation |
| Missing | #argpairs not extracted holding a relation |
| Generated | #argpairs extracted |
| **Labeled argument detection, domain and range filtering** | |
| Correct | #argpairs holding a relation and with the right candidate assigned |
| Spurious | #argpairs for which no relation holds and candidates are assigned |
| Missing | #argpairs which hold a relation and either all assigned candidates are wrong or no candidate is assigned |
| Generated | #argpairs with at least one candidate assigned |
| **Classifier** | |
| Correct | #argpairs whose highest confidence candidate is the right relation |
| Spurious | #argpairs for which no relation holds and the highest confidence candidate is wrong |
| Missing | #argpairs which hold a relation and either the highest confidence candidate is wrong or no candidate is predicted |
| Generated | #argpairs with at least one candidate predicted |
| **Conflict Resolution** | |
| Correct | #relations correct |
| Spurious | #relations whose arguments do not hold any relation |
| Missing | #relations whose arguments hold a different relation plus valid relations whose arguments are not assigned any relation. |
| Generated | #relations outputted |

Table 3: Performance measures per module.

### 5.2. Performance per module

In this section, we evaluate the overall performance after each module (Table 4). This is useful to perform error analysis, since the number of errors made and fixed by each module can be quantified. Performance at each module is calculated using the measures detailed in Table 3. Out of the eight modules (Section 4), it is only useful to calculate performance per module after argument detection, domain and range filtering, classifiers and conflict resolution: preprocessing only uses external tools to annotate input text, the bracketer modifies the parse tree but otherwise has no effect, grouping deterministically clusters argument pairs into the nine generic patterns and feature extraction only extracts values for features to feed the classifier.

| | | Unlabeled ArgDet | Labeled ArgDet | D / R Filtering | Classifier | Conflict Res. |
|---|---|---|---|---|---|---|
| Correct | | 3,579 | 2,986 | 2,861 | 2,727 | 2,619 |
| Errors | Spurious | 9,767 | 9,319 | 8,341 | 8,341 | 5,156 |
| | Missing | 5 | 598 | 723 | 857 | 1,019 |
| | Total | 9,772 | 9,917 | 9,064 | 9,198 | 6,175 |
| Generated | | 13,348 | 12,873 | 11,807 | 11,807 | 8,160 |
| Errors pruned | | n/a | −145 | 853 | −134 | 3,023 |
| Annotated | | 3,584 | 3,584 | 3,584 | 3,584 | 3,584 |
| Precision | | 0.268 | 0.232 | 0.242 | 0.231 | 0.321 |
| Recall | | 0.999 | 0.833 | 0.798 | 0.761 | 0.731 |
| F-measure | | 0.423 | 0.363 | 0.372 | 0.354 | 0.446 |

Table 4: Performance analysis per module.

Note that both labeled argument detection, and domain and range filtering have the same definition for the performance measures. This is because the latter module filters the candidates proposed by the former, but otherwise output the same kind of information (argument pairs and candidates). Also, note that it is not necessarily the case that $generated = correct + spurious$. For example, when evaluating labeled argument detection, an extracted argument pair that holds a relation and to which all assigned candidates are wrong is counted as $generated$ and $missing$.

Errors are divided into *spurious* and *missing*. In general, one can think of *spurious* errors as what was overgenerated and *missing* errors as what was ignored; the specific definitions depend on the module we are evaluating (Table 3). *Errors pruned* always refers to the difference in total errors between the current and previous module, a positive number indicates that errors are pruned and a negative number that errors are introduced. Precision, recall and f-measure are always calculated as follows:

$$precision = \frac{correct}{generated}$$

$$recall = \frac{correct}{annotated}$$

$$f\text{-}measure = \frac{2 \times precision \times recall}{precision + recall}$$

Unlabeled argument detection extracts most argument pairs holding a relation (recall 0.999), but also a lot of pairs that do not hold any relation (precision 0.268, approximately 3 out of 4 argument pairs extracted do not encode a relation). Out of the 3,579 correct argument pairs extracted, labeled argument detection assigns the right candidate for the pair (along with other candidates) to 2,986 pairs. This is responsible for a decrease in recall of 0.166, from 0.999 to 0.833. Domain and range filtering discards candidates assigned to argument pairs by discarding candidate relations whose definition are not compatible with the arguments' semantic classes. This process reduces the number of argument pairs generated from 12,873 to 11,807 (−1,066, if all candidates are filtered out the argument pair is not counted as generated (Table 3)). This module is not perfect, for 125 pairs the right candidate is discarded ($LabeledArgDet_{correct}$ = 2,986, $DRFiltering_{correct}$ = 2,861, 2,986 − 2,861 = 125). However, the vast majority of candidate relations discarded do not hold and the module significantly improves overall performance (Section 5.1).

Out of the 2,861 argument pairs with the correct candidate selected after domain and range filtering, the classifiers pick up the right candidate for 2,727 pairs. However, the majority of argument pairs (8,341) at this point are spurious (i.e., they do not hold any relation and yet the classifiers assign a relation to them), but the classifiers are unable to predict that no relation holds between these pairs.

Finally, the vast majority of relations discarded by conflict resolution are incorrect. This module decreases recall from 0.761 to 0.731 (−0.030), but the increase in precision from 0.231 to 0.321 (0.090) makes it very valuable. Overall, conflict resolution reduces the total errors by 3,023. Note that the spurious errors (relations correctly discarded) are reduced by 3,185, but 162 missing errors are introduced (relations wrongly discarded).

## 6. Discussion and Future Work

Polaris extracts relations from text and its output is provided in RDF, a standard for automatic reasoning and knowledge interchange. It is used at Lymba for a variety of real-world applications with high demand in industry, such as ontology creation, advanced question answering from heterogeneous sources of data, textual entailment recognition, and others. APIs have been developed to easily interact with Polaris from other applications. Currently, Polaris takes about 1 second to process 5 KB of free English text.

Some issues remain unsolved and several improvements are scheduled. We plan to incorporate semantic primitives and expect an improvement similar to the one brought by domain and range filtering. Incorporating primitives will fully integrate our proposal for an extended definition of semantic relations (Blanco and Moldovan, 2011).

Composing the relations provided by Polaris is another addition. Our framework for composing relations (Blanco and Moldovan, 2011) will facilitate the extraction of relations between concepts that are far away in a sentence which normally are not considered by a semantic parser, and we believe it will bring an improvement against the benchmark. Finally, we also plan to add an extra module to customize the relation inventory using inference axioms without modifying the current Polaris implementation.

## 7. References


James F. Allen, Mary Swift, and Will de Beaumont. 2008. Deep Semantic Analysis of Text. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 343–354. College Publications.

Mithun Balakrishna, Dan Moldovan, Marta Tatu, and Marian Olteanu. 2010. Semi-Automatic Domain Ontology Creation from Text Resources. In *Proceedings of the Seventh International Language Resources and Evaluation (LREC'10)*.

Eduardo Blanco and Dan Moldovan. 2011. Unsupervised Learning of Semantic Relation Composition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*, Portland, OR, USA.

Johan Bos. 2008. Wide-Coverage Semantic Analysis with Boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 277–286. College Publications.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: semantic role labeling. In *CONLL '05: Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164, Morristown, NJ, USA. Association for Computational Linguistics.

Mona Diab, Musa Alkhalifa, Sabry ElKateb, Christiane Fellbaum, Aous Mansouri, and Martha Palmer. 2007. SemEval-2007 Task 18: Arabic Semantic Labeling. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 93–98, Prague, Czech Republic, June. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling Of Semantic Roles. *Computational Linguistics*, 28:245–288.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 13–18, Prague, Czech Republic, June. Association for Computational Linguistics.

Roxana Girju. 2003. Automatic Detection of Causal Relations for Question Answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003), Workshop on "Multilingual Summarization and Question Answering - Machine Learning and Beyond"*.

Hermann Helbig. 2005. *Knowledge Representation and the Semantics of Natural Language*. Springer, 1st edition.

Iris Hendrickx, Su N. Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 94–99, Boulder, Colorado, June. Association for Computational Linguistics.

Lluís Màrquez, Lluis Villarejo, M. A. Martí, and Mariona Taulé. 2007. SemEval-2007 Task 09: Multilevel Semantic Annotation of Catalan and Spanish. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 42–47, Prague, Czech Republic, June. Association for Computational Linguistics.

Dan Moldovan and Adriana Badulescu. 2005. A Semantic Scattering Model for the Automatic Interpretation of Genitives. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 891–898, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Dan Moldovan, Marta Tatu, and Christine Clark. 2010. Role of Semantics in Question Answering. In Phillip Sheu, Heather Yu, C. V. Ramamoorthy, Arvind K. Joshi, and Lotfi A. Zadeh, editors, *Semantic Computing*. Wiley-IEEE Press, May.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised Semantic Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore, August. Association for Computational Linguistics.

James Pustejovsky and Marc Verhagen. 2009. SemEval-2010 Task 13: Evaluating Events, Time Expressions, and Temporal Relations (TempEval-2). In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 112–116, Boulder, Colorado, June. Association for Computational Linguistics.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2009. SemEval-2010 Task 10: Linking Events and Their Participants in Discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 106–111, Boulder, Colorado, June. Association for Computational Linguistics.

Marta Tatu and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 371–378, Morristown, NJ, USA. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2010. A Taxonomy, Dataset, and Classifier for Automatic Noun Compound Interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 678–687, Uppsala, Sweden, July. Association for Computational Linguistics.