

# An Inference Model for Semantic Entailment in Natural Language

Rodrigo de Salvo Braz    Roxana Girju    Vasin Punyakanok

Dan Roth    Mark Sammons

Department of Computer Science  
University of Illinois at Urbana-Champaign  
Urbana, IL, 61801, USA

{braz, girju, punyakan, danr, mssammon}@cs.uiuc.edu

*Semantic entailment* is the problem of determining if the meaning of a given sentence entails that of another. This is a fundamental problem in natural language understanding that provides a broad framework for studying language variability and has a large number of applications. We present a principled approach to this problem that builds on inducing representations of text snippets into a hierarchical knowledge representation along with a sound inferential mechanism that makes use of it to prove semantic entailment.

## 1 General Description of Our Approach

Given two text snippets  $S$  (source) and  $T$  (target) (typically, but not necessarily,  $S$  consists of a short paragraph and  $T$ , a sentence) we want to determine if  $S \models T$ , which we read as “ $S$  entails  $T$ ” and, informally, understand to mean that *most people would agree that the meaning of  $S$  implies that of  $T$* . Somewhat more formally, we say that  $S$  entails  $T$  when some representation of  $T$  can be “matched” (modulo some meaning-preserving transformations to be defined below) with some (or part of a) representation of  $S$ , at some level of granularity and abstraction.

The approach consists of these components:

**KR:** A Description Logic based hierarchical knowledge representation, EFDL, into which we represent the surface level text representations, augmented with induced syntactic and semantic parses and word and phrase level abstractions.

**KB:** A knowledge base consisting of syntactic and semantic rewrite rules, written in EFDL.

**Subsumption:** An extended subsumption algorithm which determines subsumption between EFDL expressions (representing text snippets or rewrite rules). “Extended” here means that the basic

unification operator is extended to support several word level and phrase level abstractions.

First a set of machine learning based resources are used to induce the representation for  $S$  and  $T$ . The entailment algorithm then proceeds in two phases: (1) it incrementally generates new representations of the original representation of the source text  $S$  and (2) it makes use of an (extended) subsumption algorithm to check whether any of the alternative representations of the source entails the representation of the target  $T$ . The subsumption algorithm is used in both phases in slightly different ways.

Figure 1 provides an example of the representation of two text snippets along with a sketch of the extended subsumption to decide the entailment.

## 2 Hierarchical Knowledge Representation

Our semantic entailment approach relies heavily on a hierarchical representation of natural language sentences, defined formally over a domain  $\mathcal{D} = \langle \mathcal{V}, \mathcal{A}, \mathcal{E} \rangle$  which consists of a set  $\mathcal{V}$  of typed elements, a set  $\mathcal{A}$  of attributes of elements, and a set  $\mathcal{E}$  of relations among elements. We use a Description-Logic inspired language, *Extended Feature Description Logic (EFDL)*, an extension of (Cumby and Roth, 2003). As described there, expressions in the language have an equivalent representation as *concept graphs*, and we refer to the latter representation here for comprehensibility.

*Nodes* in the concept graph represent elements – words or (multiple levels of) phrases. *Attributes* of nodes represent properties of elements. Examples of attributes include {LEMMA, WORD, POS, MAINVERB, PHTYPE, PHHEAD, NETYPE, SRLTYPE {ARG0, . . . ARGM}, NEG}. The first three are word

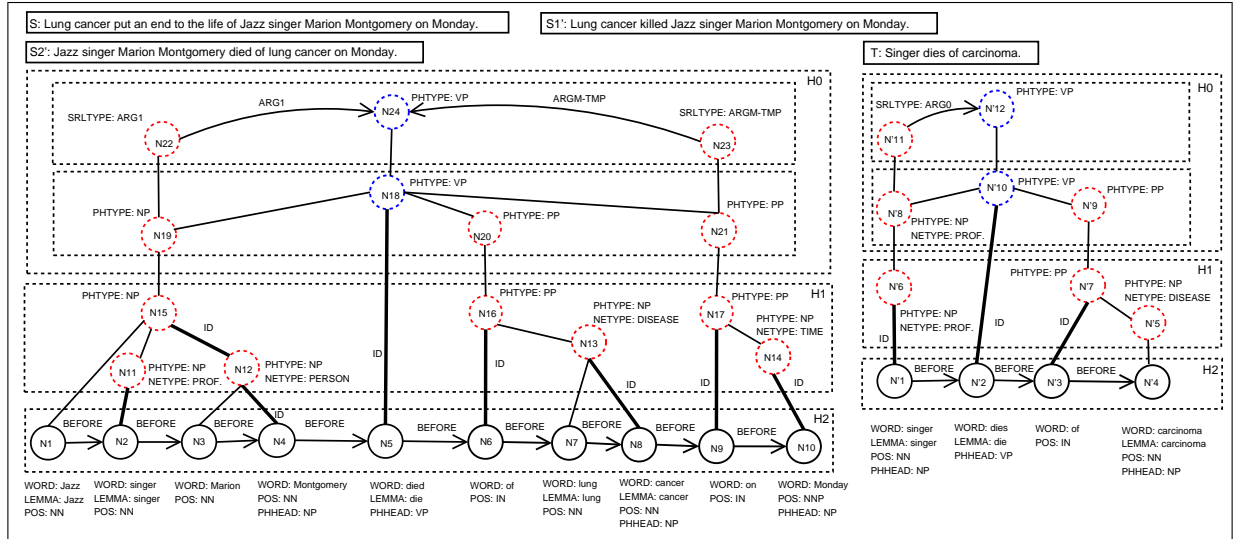


Figure 1: Example of *Re-represented Source & Target* pairs as concept graphs. The original source sentence  $S$  generated several alternatives including  $S'_1$  and the sentence in the figure ( $S'_2$ ). Our algorithm was not able to determine entailment of the first alternative (as it fails to match in the extended subsumption phase), but it succeeded for  $S'_2$ . The dotted nodes represent phrase level abstractions.  $S'_2$  is generated in the first phase by applying the following chain of inference rules: #1 (genitives): “Z’s W  $\rightarrow$  W of Z”; #2: “X put end to Y’s life  $\rightarrow$  Y die of X”. In the extended subsumption, the system makes use of WordNet hypernymy relation (“lung cancer” IS-A “carcinoma”) and NP-subsumption rule (“Jazz singer Marion Montgomery” IS-A “singer”). The rectangles encode the hierarchical levels ( $H_0, H_1, H_2$ ) at which we applied the extended subsumption. Also note that in the current experiments we don’t consider noun plurals and verb tenses.

level, the next three are phrase level, NETYPE is the named entity of a phrase, SRLTYPE is the set of semantic arguments as defined in PropBank (Kingsbury et al., 2002) and NEG is a negation attribute.

*Relations* (roles) between two elements are represented by labeled edges between the corresponding nodes. Examples of roles include: {BEFORE, ID, MOD, {ARG0, ... ARGM}}; BEFORE indicates the order between two individuals, ID and MOD represent a *contains* relation between a word and a phrase where the word, respectively, is or is not the head.

Concept graphs are used both to describe sentence representations and rewrite rules. Details are omitted here; we just mention that the expressivity of these differ - the body and head of rules are simple chain graphs, for inference complexity reasons. Restricted expressivity is an important concept in Description Logics (Baader et al., 2003), from which we borrow several ideas and nomenclature.

Concept graph representations are induced via state of the art machine learning based resources that include a tokenizer, a lemmatizer, a part-of-speech tagger, a syntactic parser, a semantic parser, a named entity recognizer, and a name coreference system.

Rewrite rules were filtered from a large collection of paraphrase rules developed in (Lin and Pantel, 2001) and compiled into our language; a number of non-lexical rewrite rules were generated manually. Currently, our knowledge base consists of approximately 300 inference rules.

The most significant aspect of our knowledge representation is its **hierarchy**. It is defined over a set of typed elements that are partitioned into several classes in a way that captures levels of abstraction and is used by the inference algorithm to exploit these inherent properties of the language. The hierarchical representation provides flexibility – rewrite rules can depend on a level higher than the lexical one, as in: [W/PHTYPE=NP] of [Z/PHTYPE=NP]  $\rightarrow$  Z’s W. Most importantly, it provides a way to abstract over variability in natural language by supporting inference at a higher than word level, and thus supports the inference process in recovering from inaccuracies in lower level representations. Consider the following example in which processing at the semantic parse level exhibits identical structure, despite significant lexical level differences.

S: “[The bombers]/A0 managed [to enter [the embassy build-

ing]/A1]/A1.”<sup>1</sup>

T: “[The terrorists]/A0 entered [the edifice]/A1.”

On the other hand, had the phrase *failed to enter* been used instead of *managed to enter*, a NEG attribute associated with the main verb would prevent this inference. Note that failure of the semantic parser to identify the semantic arguments A0 and A1 will not result in a complete failure of the inference, as described in the next section: it will result in a lower score at this level that the optimization process can compensate for (in the case that lower level inference occurs).

### 3 Inference Model and Algorithm

An exact subsumption approach that requires the representation of  $T$  be entirely embedded in the representation of  $S'_i$  is unrealistic. Natural languages allow words to be replaced by synonyms, modifier phrases to be dropped, etc., without affecting meaning. We define below our notion of extended subsumption, computed given two representations, which is designed to exploit the hierarchical representation and capture multiple levels of abstractions and granularity of properties represented at the sentence, phrase, and word-level.

Nodes in a concept graph are grouped into different hierarchical sets denoted by  $H = \{H_0, \dots, H_j\}$  where a lower value of  $j$  indicates higher hierarchical level (more important nodes). This hierarchical representation is derived from the underlying concept graph and plays an important role in the definitions below. We say that  $S'_i$  entails  $T$  if  $T$  can be *unified into*  $S'_i$ . The significance of definitions below is that we define unification so that it takes into account both the hierarchical representation and multiple abstractions.

Let  $V(T)$ ,  $E(T)$ ,  $V(S'_i)$ , and  $E(S'_i)$  be the sets of nodes and edges in  $T$  and  $S'_i$ , respectively. Given a hierarchical set  $H$ , a *unification* is a 1-to-1 mapping  $U = (U_V, U_E)$  where  $U_V : V(T) \mapsto V(S'_i)$ , and  $U_E : E(T) \mapsto E(S'_i)$  satisfying:

1.  $\forall (x, y) \in U : x$  and  $y$  are in the same hierarchical level.

2.  $\forall (e, f) \in U_E : e$  and  $f$  must be unified accordingly. That is, for  $n_1, n_2, m_1$ , and

$m_2$  which are the sinks and the sources of  $e$  and  $f$  respectively,  $(n_1, m_1) \in U_V$  and  $(n_2, m_2) \in U_V$ .

Let  $\mathcal{U}(T, S'_i)$  denote the space of all unifications from  $T$  to  $S'_i$ . In our inference, we assume the existence of a unification function  $G$  determining the cost of unifying pairs of nodes or pairs of edges.  $G$  may depend on language and domain knowledge, e.g. synonyms, name matching, and semantic relations. When two nodes or edges cannot be unified,  $G$  returns infinity. This leads to the definition of *unifiability*.

**Definition 3.1** Given a hierarchical set  $H$ , a unification function  $G$ , and two concept graphs  $S'_i$  and  $T$ , we say that  $T$  is unifiable to  $S'_i$  if there exists a unification  $U$  from  $T$  to  $S'_i$  such that the cost of unification defined by

$$D(T, S'_i) = \min_{U \in \mathcal{U}(T, S'_i)} \sum_{H_j} \sum_{(x, y) \in U | x, y \in H_j} \lambda_j G(x, y)$$

is finite, where  $\lambda_j$  are some constants s.t. the cost of unifying nodes at higher levels dominates those of the lower levels.

Because top levels of the hierarchy dominate lower ones, nodes in both graphs are checked for subsumption in a top down manner. The levels and corresponding processes are:

Hierarchy set  $H_0$  corresponds to sentence-level nodes, represented by the verbs in the text. The inherent set of attributes is {PHTYPE, MAINVERB, LEMMA}. In order to capture the argument structure at sentence-level, each verb in  $S'_i$  and  $T$  has a set of edge attributes {ARG $_i$ , PHTYPE $_i$ }, where ARG $_i$  and PHTYPE $_i$  are the semantic role label and phrase type of each argument  $i$  of the verb considered.

For each verb in  $S'_i$  and  $T$ , check if they have the same attribute set and argument structure at two abstraction levels:

1) The semantic role level (SRL attributes). eg: ARG0 verb ARG1 : [Contractors]/ARG0 build [houses]/ARG1 for \$100,000.

2) The syntactic parse level (parse tree labels). Some arguments of the verb might not be captured by the semantic role labeler (SRL); we check their match at the syntactic parse level. eg: NP verb NP PP : [Contractors]/NP build [houses]/NP [for \$100,000]/ PP.

At this level, if all nodes are matched (modulo functional subsumption), the cost is 0, otherwise it is infinity.

<sup>1</sup>The verbs “manage” and “enter” share the semantic argument “[the bombers]/A0”.

Hierarchy set  $H_1$  corresponds to phrase-level nodes and represents the semantic and syntactic arguments of the  $H_0$  nodes (verbs). If the phrase-level nodes are recursive structures, all their constituent phrases are  $H_1$  nodes. For example, a complex noun phrase consists of various base-NPs. Base-NPs have edges to the words they contain.

The inference procedure recursively matches the corresponding  $H_1$  nodes in  $T$  and  $S'_i$  until it finds a pair whose constituents do not match. In this situation, a *Phrase-level Subsumption* algorithm is applied. The algorithm is based on subsumption rules that are applied in a strict order (as a decision list) and each rule is assigned a confidence factor.

The algorithm makes sure two  $H_1$  nodes have the same PHTYPE, but allows other attributes such as NETYPE to be optional. Each unmatched attribute results in a uniform cost.

Hierarchy set  $H_2$  corresponds to word-level nodes. The attributes used here are: {WORD, LEMMA, POS}. Unmatched attributes result in a uniform cost.

We solve the subsumption problem by formulating an equivalent Integer Linear Programming (ILP) problem of which details is omitted. Despite the fact that this optimization problem is NP hard, commercial packages have very good performance on sparse problems such as this one (Xpress-MP, ).

## 4 Experimental Evaluation

The system needs to establish a confidence threshold to decide, for each example, if the entailment is true or false. The system searches for the optimal threshold per task on the development set. It took about 50 minutes to run the system on the development set and 2 hours on the test. Table 1 shows the system’s accuracy on the development set.

	All	Task						
		CD	IE	IR	MT	PP	QA	RC
System	64.8	74.0	35.0	62.0	87.5	63.8	84.0	49.0
Test	56.1	77.3	50.0	52.2	53.3	50.0	50.0	51.4

Table 1: System’s performance obtained for each experiment on the Pascal corpora and its subtasks.

Below are three examples highlighting some interesting aspects of the entailment system. In the first example, the system fails to produce the correct answer due to its inability to identify the verbal

paraphrase in the long sentence. However, it successfully captures some hard entailment pairs such as those in examples 2 and 3.

S1: “As oil prices soared to new heights after a terrorist attack in Saudi Arabia, the nation’s influential oil minister tried to reassure markets yesterday that OPEC would do its best to provide adequate supplies.”

T1: “As oil prices soared after a terrorist attack in Saudi Arabia, the nation’s oil minister tried to reassure markets that OPEC will try to provide adequate supplies.”

S2: “A male gorilla escaped from his cage in the Berlin zoo and sent terrified visitors running for cover, the zoo said yesterday.”

T2: “A gorilla escaped from his cage in a zoo in Germany.”

S3: “The recent G8 summit, which was first held on July 13-16, 1975, took place on Sea Island on June 8-10.”

T3: “The recent G8 summit took place on July 13-16, 1975.”

In the second example, the system takes advantage of the functional subsumption and identifies the PART-OF semantic relation between “Berlin” and “Germany”. The system correctly classifies the third pair as negative as it fails to match the date after it passes the argument structure test.

## Acknowledgement

We thank Dash Optimization for the free academic use of their Xpress-MP software. This work was supported by the Advanced Research and Development Activity (ARDA)’s Advanced Question Answering for Intelligence (AQUAINT) Program, NSF grant ITR-IIS-0085980, and ONR’s TRECC and NCASSR programs.

## References

- F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. 2003. *Description Logic Handbook*. Cambridge.
- C. M. Cumby and D. Roth. 2003. Learning with feature description logics. In S. Matwin and C. Sammut, editors, *The 12th Intl. Conference on Inductive Logic Programming (ILP)*. Springer. LNAI 2583.
- P. Kingsbury, M. Palmer, and M. Marcus. 2002. Adding semantic annotation to the Penn treebank. In *Proceedings of the Human Language Technology conference (HLT)*, San Diego, CA.
- D. Lin and P. Pantel. 2001. DIRT: discovery of inference rules from text. In *KDD '01*, pages 323–328.
- Xpress-MP. Dash Optimization. Xpress-MP. <http://www.dashoptimization.com/products.html>.