

# Knowledge Representation for Semantic Entailment and Question-Answering

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

## Abstract

*Semantic entailment* is the problem of determining if the meaning of a given sentence entails that of another. *Question-answering* can be reduced to this problem by rephrasing the question as a statement that is entailed by correct answers. In [Braz *et al.*, ] we present a principled approach to semantic entailment that builds on inducing re-representations of text snippets into a hierarchical knowledge representation along with an optimization-based inferential mechanism that makes use of it to prove semantic entailment.

This paper provides details and analysis of the knowledge representation and knowledge resources issues in the above approach. We analyze our system's behavior on a collection of question-answer pairs and use it to motivate and explain some of the design decisions in our hierarchical knowledge representation, that is centered around a predicate-argument type abstract representation of text.

## 1 Introduction

*Semantic entailment* is the task of determining, for example, that the sentence: “WalMart defended itself in court today against claims that its female employees were kept out of jobs in management because they are women” entails that “Wal-Mart was sued for sexual discrimination”.

Determining whether the meaning of a given text snippet entails that of another or whether they have the same meaning is a fundamental problem in natural language understanding that requires the ability to abstract over the inherent syntactic and semantic variability in natural language [Dagan and Glickman, 2004]. This challenge is at the heart of many high level natural language processing tasks including Question Answering, Information Retrieval and Extraction, Machine Translation, and others that attempt to reason about and capture the meaning of linguistic expressions.

Research in natural language processing in the last few years has concentrated on developing resources that provide multiple levels of syntactic and semantic analysis, resolve context sensitive ambiguities, and identify relational structures and abstractions (from syntactic categories like POS tags to semantic categories such as named entities).

However, beyond these resources, in order to support fundamental tasks such as inferring semantic entailment between two text snippets, there needs to be a unified knowledge representation of the text that (1) provides a hierarchical encoding of the structural, relational and semantic properties of the given text, (2) is integrated with learning mechanisms that can be used to induce such information from raw text, and (3) is equipped with an inferential mechanism that can be used to support inferences over such representations.

Relying on general purpose knowledge representations — FOL, probabilistic or hybrids — along with their corresponding general purpose inference algorithms does not resolve the key issues of *what to represent* and *how to derive a sufficiently abstract representation* and, in addition, may lead to brittleness and complexity problems. On the other hand, relying only on somewhat immediate correspondences between question and candidate answers, such as shared words or shared named entities, has strong limitations. We avoid some of these problems by *inducing* an abstract representation of the text which does not attempt to represent the full meaning of text, but provides what could be seen as a *shallow semantic* representation; yet, it is significantly more expressive than extraction of straightforward phrase-level characteristics. We induce this into a description-logic based language that is more restricted than FOL yet is expressive enough to allow both easy incorporation of language and domain knowledge resources and strong inference mechanisms.

Unlike traditional approaches to inference in natural language [Schubert, 1986; Moore, 1986; Hobbs *et al.*, 1988] our approach (1) makes use of *machine learning* based resources in order to induce an abstract representation of the input data, as well as to support multiple inference stages and (2) models inference as an *optimization* process that provides robustness against inherent variability in natural language, inevitable noise in inducing the abstract representation, and missing information.

In this paper, we focus on the hierarchical knowledge representation used by our system. We also present a brief explanation of the inference algorithm — described in detail in a companion paper [Braz *et al.*, ] — and a detailed experimental analysis of our system that highlights the advantages of the hierarchical approach. Along with the formal definition and justification developed here for our computational approach to semantic entailment, our knowledge representation and al-

gorithmic approach provide a novel solution that addresses some of the key issues the natural language research community needs to resolve in order to move forward towards higher level tasks of this sort. Namely, we provide ways to represent knowledge, either external or induced, at multiple levels of abstractions and granularity, and reason with it at the appropriate level. The preliminary evaluation of our approach is very encouraging and illustrates the significance of some of its key contributions.

## 1.1 General Description of Our Approach

We reduce the problem of question answering to that of textual semantic entailment. Whether a candidate text actually answers a question can be inferred by deciding if the question, converted to a statement with a placeholder, is entailed by the candidate answer. For example, “John bought the book yesterday downtown” is a correct answer to the question “Who bought the book?” because it entails “XXX bought the book” (the placeholder can be thought of as an existentially quantified variable).

Specifically, given two text snippets  $S$  (source) and  $T$  (target) where typically, but not necessarily,  $S$  consists of a short paragraph and  $T$ , a sentence, textual semantic entailment is the problem of determining if  $S \models T$ , which we read as “ $S$  entails  $T$ ”. This informally means that *most people would agree that the meaning of  $S$  implies that of  $T$* . 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 the following components:

**KR:** A Description Logic based hierarchical knowledge representation, EFDL (Extended Feature Description Logic), [Cumby and Roth, 2002], into which we re-represent the surface level text, 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 re-representations of the original representation of the source text  $S$  by augmenting it with heads of subsumed rewrite rules, and (2) it makes use of an optimization based (extended) subsumption algorithm to check whether any of the alternative representations of the source entails the representation of the target  $T$ . The extended subsumption algorithm is used both in checking final entailment and in determining when and how to generate a re-representation in slightly different ways.

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

## 2 Hierarchical Knowledge Representation

Our abstract representation of text passages is based on a predicate-argument structure at both semantic and syntactic levels. The semantic representation captures predicate-argument relations following the PropBank [Kingsbury *et al.*, 2002] representation. PropBank is a semantically annotated version of the Wall Street Journal portion of Penn Treebank and provides consistent semantic role labels across different syntactic realizations of the same verb as shown in the following example:

[Mary]/ARG0 left [the room]/ARG1.

Here, ARG0 represents the *leaver* and ARG1 the *thing left* as arguments of the verb *leave*.

Currently, we focus only on verb-argument relations, but our representation can be easily extended to other types of predicates, such as nouns, adjectives, and adverbs. This representation is induced by a machine learning based Semantic Role Labeler ([Punyakanok *et al.*, 2004]) that identifies the verb’s arguments and semantically annotates them with corresponding PropBank labels in context.

At the syntactic level, the representation captures full parse information for the text considered. For this, we rely on Collin’s parser [Collins, 1999]. For example,

[Mary]/NP left [the room]/NP/ [in a hurry]/PP,

where the two noun phrases and the prepositional phrase are attached to the verb *left*.

Besides these, we also consider a word-level representation. Each word is annotated with various information such as part-of-speech and lemma.

The abstract representation is also enriched with other types of knowledge, such as named entities (eg, “John Smith” is a PERSON), qualifier labels (eg, “some people”, “no children”), negation (eg, “didn’t succeed”), modality (eg, “could”, “might”), temporal information (eg, “after an event”, “before an action”), and coreference (both pronoun and name coreference).

The most significant aspect of our knowledge representation is its **hierarchy**. It captures the semantic, syntactic, and lexical levels of abstraction thus described and is used by the inference algorithm to exploit these inherent properties of the language. The hierarchical representation provides flexibility as the source and target snippets can be matched at the corresponding level. Most importantly, it provides a way to abstract over variability in natural language by supporting inference at a higher than word level, and thus also supports the inference process in recovering from inaccuracies in inducing the representation. Consider, for example, the following pair of sentences, in which processing at the semantic parse level exhibits identical structure, despite significant lexical level differences.

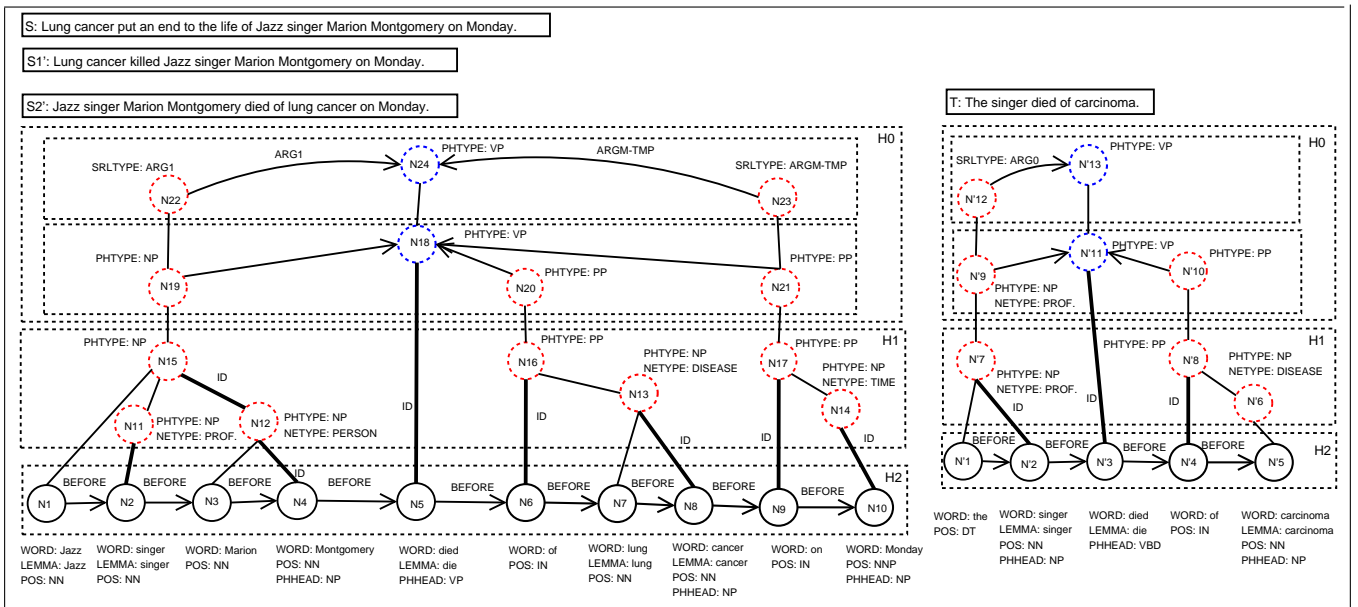


Figure 1: Example of *Re-represented Source & Target* pairs as concept graphs. The original source sentence  $S$  generated several alternatives including  $S'_1$  and the sentence in the figure ( $S'_2$ ). Our algorithm was not able to determine entailment of the first alternative (as it fails to match 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 in the extended subsumption, although our system has this capability.

S: “[The bombers]/r/ARG0 managed [to enter [the embassy building]/ARG1]/ARG1.”<sup>1</sup>

T: “[The terrorists]/ARG0 entered [the edifice]/ARG1.”

On the other hand, had the phrase *failed to enter* been used instead of *managed to enter*, a negation attribute associated with the main verb would prevent this inference. Note that failure of the semantic parser to identify the semantic arguments ARG0 and ARG1 will not result in a complete failure of the inference, as described in the inference 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).

In the following subsection we present a formal description of the knowledge representation.

## 2.1 Formal Description of The Knowledge Representation

The 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 FDL [Cumby and

<sup>1</sup>The verbs “manage” and “enter” share the semantic argument “[the bombers]/ARG0”.

Roth, 2002]. 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 (they are explained in more detail later) include {LEMMA, WORD, POS, PREDICATE.VALUE, PHTYPE, PHHEAD, NETYPE, ARGTYPE, NEGATION}. The first three are word level, the next three are phrase level, NETYPE is the named entity of a phrase, ARGTYPE is the set of semantic arguments as defined in PropBank [Kingsbury *et al.*, 2002] and NEGATION is a negation attribute. Only attributes with non-null values need to be specified.

*Relations* (roles) between two elements are represented by labeled edges between the corresponding nodes. Examples of roles (again, explained in more detail later) include: {BEFORE, ARG0, ... ARG5}; BEFORE indicates the order between two individuals, and ARG0 represents the relations between a predicate (verb) and its argument.

Figure 1 shows a visual representation of a pair of sentences rerepresented as concept graphs.

Concept graphs are used both to describe instances (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 impor-

tant 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 a part-of-speech tagger [Even-Zohar and Roth, 2001], a syntactic parser [Collins, 1999], a semantic parser [Punyakanok *et al.*, 2004; 2005], a named entity recognizer<sup>2</sup>, and a name coreference system [Li *et al.*, 2004] with the additional tokenizer and lemmatizer derived from WordNet [Fellbaum, 1998]. 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.

### Rule representation

A rule is a pair  $(lhs, rhs)$  of concept graphs ( $lhs$  is the rule’s *body*, while  $rhs$  is its *head*). These concept graphs are restricted in that they must be *paths*. This restricts the complexity of the inference algorithm while keeping them useful enough for our purposes. As a shorthand, more than one path can be described in  $lhs$ , but in this case the rule is implicitly treated as more than one rule, each with one path and the same  $rhs$ .

$lhs$  describes a structure to match in the sentence concept graph, while  $rhs$  describes a new predicate (and related attributes and edges) to be added to the sentence concept graph in case there is a match.  $rhs$  can also describe attributes to add to one or more existing nodes without adding a new predicate, provided no new edges are introduced. These restrictions ensure that the data representation always remains, from the subsumption algorithm’s perspective, a set of overlapping trees.

Variables can be used in  $lhs$  so that we can specify which entities have edges/attributes added by  $rhs$ . Rules thus allow rewrite of (part of) original sentence; e.g. we can encode DIRT [Lin and Pantel, 2001] rules as predicate/argument structures and use them to allow (parts of) the original sentence to be re-represented via paraphrase, by linking existing arguments with new predicates.

## 3 Algorithmic Semantic Entailment

This section follows the presentation in [Braz *et al.*, ] and formally defines and justifies our algorithmic approach to semantic entailment.

Let  $\mathcal{R}$  be a knowledge representation language with a well defined syntax and semantics over a domain  $\mathcal{D}$ . Specifically, we think of elements in  $\mathcal{R}$  as expressions in the language or, equivalently, as the set of interpretations that satisfy it [Lloyd, 1987]. Let  $r$  be a mapping from a set of text snippets  $\mathcal{T}$  to a set of expressions in  $\mathcal{R}$ . Denote the images of two text snippets  $S, T$ , under this mapping by  $r_S, r_T$ , respectively. Given the set of interpretations over  $\mathcal{D}$ , let  $M$  be a mapping from an expression in  $\mathcal{R}$  to the corresponding set of interpretations it satisfies. For expressions  $r_S, r_T$ , the images

<sup>2</sup>Named entity recognizer from Cognitive Computation Group, <http://l2r.cs.uiuc.edu/~cogcomp>

of  $S, T$  under  $\mathcal{R}$ , their model theoretic representations thus defined are denoted  $M(r_S), M(r_T)$ .

Conceptually, as in the traditional view of semantic entailment, this leads to a well defined notion of entailment, formally defined via the model theoretic view; traditionally, the algorithmic details are left to a *theorem prover* that uses the syntax of the representation language, and may also incorporate additional knowledge in its inference. We follow this view, and use a notion of *subsumption* between elements in  $\mathcal{R}$ , denoted  $u \sqsubseteq v$ , for  $u, v \in \mathcal{R}$ , that is formally defined via the model theoretic view – when  $M(u) \subseteq M(v)$ . Subsumption between representations provides an implicit way to represent entailment, where additional knowledge is conjoined with the source to “prove” the target.

However, the proof theoretic approach corresponding to this traditional view is unrealistic for natural language. Subsumption is based on *unification* and requires, in order to prove entailment, that the representation of  $T$  is entirely embedded in the representation of  $S$ . Natural languages allow for words to be replaced by synonyms, for modifier phrases to be dropped, etc., without affecting meaning. An extended notion of subsumption is therefore needed which captures sentence, phrase, and word-level abstractions.

Our algorithmic approach is thus designed to alleviate these difficulties in a proof theory that is too weak for natural language. Conceptually, a weak proof theory is overcome by entertaining multiple representations that are equivalent in meaning. We provide theoretical justification below, followed by the algorithmic implications.

We say that a representation  $r \in \mathcal{R}$  is *faithful* to  $S$  if  $r$  and  $r_S$  have the same model theoretic representation, i.e.,  $M(r) = M(r_S)$ . Informally, this means that  $r$  is the image under  $\mathcal{R}$  of a text snippet with the same meaning as  $S$ .

**Definition 1** *Let  $S, T$  be two text snippets with representations  $r_S, r_T$  in  $\mathcal{R}$ . We say that  $S \models T$  (read:  $S$  semantically entails  $T$ ) if there is a representation  $r \in \mathcal{R}$  that is faithful to  $S$  and that is subsumed by  $r_T$ .*

Clearly, there is no practical way to exhaust the set of all those representations that are faithful to  $S$ . Instead, our approach searches a space of faithful representations, generated via a set of rewrite rules in our KB.

A *rewrite rule* is a pair  $(lhs, rhs)$  of expressions in  $\mathcal{R}$ , such that  $lhs \sqsubseteq rhs$ . Given a representation  $r_S$  of  $S$  and a rule  $(lhs, rhs)$  such that  $r_S \sqsubseteq lhs$ , the augmentation of  $r_S$  via  $(lhs, rhs)$  is the representation  $r'_S = r_S \wedge rhs$ .

**Claim 1** *The representation  $r'_S$  generated above is faithful to  $S$ .*

To see this, note that as expressions in  $\mathcal{R}$ ,  $r'_S = r_S \wedge rhs$ , therefore  $M(r'_S) = M(r_S) \cap M(rhs)$ . However, since  $r_S \sqsubseteq lhs$ , and  $lhs \sqsubseteq rhs$ , then  $r_S \sqsubseteq rhs$  which implies that  $M(r_S) \subseteq M(rhs)$ . Consequently,  $M(r'_S) = M(r_S)$  and the new representation is faithful to  $S$ .

The claim gives rise to an algorithm, which suggests incrementally *augmenting* the original representation of  $S$  via the rewrite rules, and computing subsumption using the “weak” proof theory between the augmented representation and  $r_T$ .

Informally, this claim means that while, in general, augmenting the representation of  $S$  with an expression  $rhs$  may restrict the number of interpretations the resulting expression has, in this case, since we only augment the representation when the left hand side  $lhs$  subsumes  $r_S$ , we end up with a re-representation that is in fact equivalent to  $r_S$ . Therefore, given a collection of rules  $\{(lhs, rhs)\}$  we can chain them, and incrementally generate faithful representations of  $S$ . Consequently, this algorithm is a sound algorithm<sup>3</sup> for semantic entailment according to Def. 1, but it is not complete. Its success depends on the size and quality of the rule set<sup>4</sup> applied in the search.

Two important notes are in order. First, since rewrite rules typically “modify” a small part of a sentence representation (see Fig. 1), the augmented representation provides also a compact way to encode a large number of possible representations. Second, note that while the rule augmentation mechanism provides a justification for an algorithmic process, in practice applying rewrite rules is somewhat more complicated. The key reason is that many rules have a large fan-out; that is, a large number of heads are possible for a given rule body. Examples include synonym rules, equivalent ways to represent names of people (e.g., John F. Kennedy and JFK), etc. We therefore implement the mechanism in two ways; one process which supports chaining well, in which we explicitly augment the representation with low fan-out rules (e.g., Passive-Active rules); and a second, appropriate to the large fan-out rules. In the latter, we abstain from augmenting the representation with the many possible heads but take those rules into account when comparing the augmented source with the target. For example, if a representation includes the expression “JFK/PER”, we do not augment it with all the many expressions equivalent to “JFK” but, when comparing it to a candidate in the target, such as “President Kennedy”, these equivalencies are taken into account. Semantically, this is equivalent to augmenting the representation. Instead of an explicit list of rules, the large fan-out rules are represented as a functional black box that can, in principle, contain any procedure for deciding comparisons. For this reason, this mechanism is called *functional subsumption*.

The resulting algorithmic approach is therefore:

(1) Once an EFDL representation for  $S$  and  $T$  is induced, the algorithm incrementally searches the EFDL rewrite rules in KB to find a rule with a body that subsumes the representation of  $S$ . In this case, the head of the rule is used to *augment* the EFDL representation of  $S$  and generate a new (equivalent) representation of  $S$ . KB consists of syntactic and semantic EFDL rewrite rules expressed at the word, syntactic and semantic categories, and phrase levels; applying them results in new representations  $S'_i$  that capture alternative ways of expressing the surface level text.

(2) Re-representation  $S'_i$ s are processed via the extended

<sup>3</sup>Soundness depends on a “correct” induction of the representation of the text; we do not address this theoretically here.

<sup>4</sup>The power of this search procedure is in the rules.  $lhs$  and  $rhs$  might be very different at the surface level, yet, by satisfying model theoretic subsumption they provide expressivity to the re-representation in a way that facilitates the overall subsumption.

subsumption algorithm against the representation of  $T$ . The notion of extended subsumption captures, just like the rewrite rules, several sentence, phrase, and word-level abstractions. The extended subsumption process is also used when determining whether a rewrite rule applies.

Rewrite rules and extended subsumption decisions take into account relational and structural information encoded in the hierarchical representation, which is discussed below. In both cases, decisions are quantified as input to an optimization algorithm that attempts to generate a “proof” that  $S$  entails  $T$ .

## 4 Inference Model and Algorithm

As natural languages allow the expression of various concepts in different ways without affecting meaning, an exact subsumption approach that requires the representation of  $T$  be entirely embedded in the representation of  $S'_i$  is unrealistic.

Extended subsumption is designed to take advantage of the hierarchical representation at various levels of abstraction at the sentence, phrase, and word-level. Thus, 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).

The inference procedure recursively matches the corresponding  $H_j$  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.

Figure 1 exemplifies the matching order between  $S'_i$  and  $T$  based on constraints imposed by the hierarchy.

We solve the subsumption problem by formulating an equivalent Integer Linear Programming (ILP) problem<sup>5</sup>. Details about the extended subsumption and the inference algorithm can be found in a companion paper [Braz *et al.*, ].

## 5 Experimental Evaluation and Discussion

We tested our approach on a collection<sup>6</sup> of question-answer pairs develop by Xerox PARC for a pilot evaluation of Knowledge-Oriented Approaches to Question Answering under the ARDA-AQUAINT program. The PARC corpus consists of 76 Question-Answer pairs annotated as “true”, “false” or “unknown” (and an indication of the type of reasoning required to deduce the label). The question/answer pairs provided by PARC are designed to test different cases of linguistic entailment. The corpus concentrates on examples of strict and plausible linguistic (lexical and constructional) inferences and indicates whether it involves some degree of background world knowledge. The focus is on inferences that can be made purely on the basis of the meaning of words and phrases. The questions are straightforward and therefore easily rewritten (by hand) into statement form. One sentence pair involving qualifiers was reordered to test qualifier subsumption.

<sup>5</sup>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, ].

<sup>6</sup>The data is available by following the data link from <http://l2r.cs.uiuc.edu/~cogcomp>.

For evaluation reasons, we used only two labels in our experiments, “true” and “false”, corresponding to “S entails T” and “S does not entail T”. The “unknown” instances were classified as “false”.

Of these 76 sentence pairs, 64 were perfectly tagged by our Semantic Role Labeler (SRL), and were used as a noise-free test set to evaluate our system.

This section describes the development of our system along two dimensions: one adds more structure and annotation of words and phrases, while the other adds semantic analysis components. We first outline the progressive development of our system from lexical level matching to the full system. Then, for each version of the system and for each semantic analysis component we give one or more examples of sentence pairs affected by the new version/component. Finally, we present a summary of the performance of each version of the system on the noise-free and noisy data sets.

As baseline we use lexical-level matching based on a bag-of-words representation with lemmatization and normalization (LLM). We use this as the starting point for our system. We then add Semantic Role Labeling, which gives us simple verb-level predicate-argument structure; this version of the system is labeled “SRL + LLM”. Finally, we add full parse and shallow parse structure, which allows the parsing of the SRL arguments into hierarchical structures with key entities as the roots and modifiers as the leaves. This version of the system, labeled “SRL + deep structure”, also uses Named Entity annotation from our Named Entity Recognizer (NER).

The system presently supports three semantic analysis modules: verb phrase compression, discourse analysis, and qualifier analysis. The different semantic analysis modules depend on different levels of structure: verb phrase compression requires word order and part of speech; discourse analysis requires full parse information and part of speech, and qualifier analysis requires full and shallow parse information and part of speech. These components are added in the above order for each version of the system that supports them.

Finally, there is a Knowledge Base module comprising rewrite rules that encode paraphrase and inference information. Most rules require SRL information, though some use only word order and the words themselves. In evaluating the LLM system with the KB enabled, only word-based rules can fire.

### A. LLM

We use the LLM as the starting point for our full entailment system. The LLM system ignores a large set of stopwords, which for certain positive sentence pairs allows entailment when the more sophisticated systems require a rewrite rule to map from the predicate in S to the predicate in T. For example: since the list of stopwords includes forms of “be”, the following sentence pair will be classified “true” by LLM, while the more sophisticated systems require a KB rule to link “visit” to “be (in)”:

S: *[The diplomat]/ARG1 visited [Iraq]/ARG1 [in September]/AM\_TMP*

T: *[The diplomat]/ARG1 was in [Iraq]/ARG2*

The SRL+LLM system will extract verb frames for “visit” in S and “was” in T, and will fail the subsumption check at

the verb level. For LLM, the only words of T that register are “diplomat” and “Iraq”, and as these are present in S, LLM will return “true”.

Of course, LLM is insensitive to small changes in wording. For the following sentence pair, LLM returns “true”, which is clearly incorrect:

S: *Legally, John could drive.*

T: *John drove.*

### B. SRL + LLM

The next version of the system first tries to match SRL annotation for S and T, and if this matches, it uses LLM to determine argument subsumption.

The advantage of SRL+LLM over LLM is evident when, for example, arguments are interchanged between two verbs. This case is not represented in the PARC dataset, but the following sentence pair gives such an example:

S: *[The president]/ARG0 said [[the diplomat]/ARG0 left [Iraq]/ARG1]/ARG1*

T: *[The diplomat]/ARG0 said [[the president]/ARG0 left [Iraq]/ARG1]/ARG1*

In this case, the non-stopwords in S and T are identical, so LLM will label this pair “true”. However, SRL will attach different ARG0s to “said” and “left”, and will therefore correctly label this sentence pair “false”.

The disadvantage of using SRL is that it generates predicate frames for some verbs that are ignored as stopwords by LLM, such as “went” in the following example:

S: *[The president]/ARG0 visited [Iraq]/ARG1 [in September]/AM\_TMP*

T: *[The president]/ARG0 went to [Iraq]/ARG1.*

Where LLM ignores “went”, SRL generates a predicate node, causing subsumption to fail at the predicate level and generating an incorrect “false” label.

In this data set, there are more instances like the second case above than like the first; the result is a drop in performance. However, the rest of this section will show that the SRL forms a crucial backbone that supports a more successful approach.

### The Verb Processing module

The Verb Processing (VP) module rewrites certain verb phrases as a single verb with additional attributes. It uses word order and Part of Speech information to identify candidate patterns and, when the verbs in the construction in the sentence match a pattern in the VP module, the verb phrase is replaced by a single predicate node with additional attributes representing modality (“CONFIDENCE”) and tense (“TENSE”).

The VP module presently recognizes modal constructions, tense constructions, and simple verb compounds of the form “VERB to VERB” (such as “manage to enter”). In each case, the first verb is compared to a list that maps verb lemmas to tenses and qualifiers; for example, “has” is recognized as a tense auxiliary and results in the attribute “TENSE: past” being added to the second verb’s node; the “has” node is then eliminated and the graph structure corrected.

In the example below, the VP recognizes the modal construction and adds the qualifying attribute “CONFIDENCE: potential” to the main verb node, “drive”:

S: *Legally, John could drive.*

T: *John drove.*

Subsumption in the SRL+LLM system then fails at the verb level, returning the correct value “false” for this sentence pair.

This module also acts as an enabler for other resources (such as the Knowledge Base). This may result in a decrease in performance when those modules are not present, as it corrects T sentences that may have failed subsumption in the SRL+LLM system because the auxiliary verb was not present in the corresponding S sentence:

S: *Bush said that Khan sold centrifuges to North Korea.*

T: *Centrifuges were sold to North Korea.*

The SRL+LLM system returns the correct answer, “false”, for this sentence pair, but for the wrong reason: SRL generates a separate predicate frame for “were” and for “sold” in T, and there is no matching verb for “were” in S.

When the VP module is added, the auxiliary construction in T is rewritten as a single verb with tense and modality attributes attached; the absence of the auxiliary verb means that SRL generates only a single predicate frame for “sold”. This matches its counterpart in S, and subsumption succeeds, as the qualifying effect of the verb “said” in S cannot be recognized without the deeper parse structure and the Discourse Analysis module.

On the PARC corpus, the net result of applying the VP module when the KB is not enabled is either no improvement or a decrease in performance, due to the specific mix of sentences. However, the importance of such a module to correctly identify positive examples becomes evident when the knowledge base is enabled, as the performance jumps significantly.

### C. SRL + Deep Structure

The next version of the system uses full and shallow parse, Named Entity and Part of Speech information to identify substructure in SRL predicate arguments. Specifically, the system identifies the key entity in each argument and modifiers such as adjectives, titles, and quantities. By default, we assume that T must be less specific than S (this is not always correct, but requires a new module to determine when a more general argument entails a more specific one, and when not).

The new system correctly labels the following example, which is incorrectly labeled by the LLM and SRL+LLM systems:

S: *No US congressman visited Iraq until the war.*

T: *Some US congressmen visited Iraq before the war.*

The new system includes the determiners “no” and “some” as modifiers of their respective entities; subsumption fails at the argument level because these modifiers don’t match.

However, the new system also makes new mistakes:

S: *The room was full of women.*

T: *The room was full of intelligent women.*

The LLM and SRL+LLM systems find no match for “intelligent” in S, and so return the correct answer, “false”. However, the SRL+deep structure system allows unbalanced T adjective modifiers, assuming that S must be more general than T, and returns “true”.

### Verb Phrase Module

Adding the Verb Phrase module results in performance changes similar to those when it is enabled with the SRL+LLM system, for the same reasons.

### Discourse Analysis Module

The Discourse Analysis (DA) module detects the effects of an embedding predicate on the embedded predicate. It uses the full parse tree to identify likely candidate structures, then compares the embedding verb to a list mapping verbs to modality (CONFIDENCE). The main distinction that is presently supported is between “FACTUAL” and a set of values that distinguish various types of uncertainty. This allows different assumptions to be supported; for example, if we wish to assume that when something is said, it is taken as truth, we can treat the CONFIDENCE value “REPORTED” as entailing “FACTUAL” and vice versa. The module attaches the appropriate CONFIDENCE attribute value to the embedded verb node; if this attribute is not matched during subsumption, subsumption fails.

The following example highlights the importance of the way an embedded predicate is affected by the embedding predicate. In this example, the predicate “Hanssen sold secrets to the Russians” is embedded in the predicate “The New York Times reported...”.

S: *“The New York Times reported that Hanssen sold FBI secrets to the Russians and could face the death penalty.”*

T: *“Hanssen sold FBI secrets to the Russians.”*

Our system identifies the following verb frames in S and T:

S-A: *“[The New York Times]/A0 reported [that Hanssen sold FBI secrets to the Russians... ]/A1”*

S-B: *“[Hanssen]/A0 sold [FBI secrets]/A1 to [the Russians]/A3”*

T-A: *“[Hanssen]/A0 sold [FBI secrets]/A1 to [the Russians]/A3”*

During preprocessing, our system detects the pattern “[VERB] that [VERB]”, and classifies the first verb as affecting the confidence of its embedded verb. The system marks the verb (predicate) “sold” in S with attribute and value “CONFIDENCE: REPORTED”. Thus the subsumption check determines that entailment fails at the verb level, because by default, verbs are given the attribute and value “CONFIDENCE: FACTUAL”, and the CONFIDENCE values of the “sold” nodes in S and T do not match. This is in contrast to LLM and SRL+LLM, both of which return the answer “true”.

The next example demonstrates that the implementation of this embedding detection is robust enough to handle a subtly different sentence pair: in this case, the sentence structure “Hanssen, who sold...” indicates that the reader should understand that it is already proven (elsewhere) that Hanssen has sold secrets.

S: “*The New York Times reported that Hanssen, who sold FBI secrets to the Russians, could face the death penalty.*”

T: “*Hanssen sold FBI secrets to the Russians.*”

Our system identifies the following verb frames in S and T (using the full parse data provided by Collins’ parser to connect “who” to “Hanssen”):

S-A: “[*The New York Times*]/A0 reported [*that Hanssen, who sold FBI secrets to the Russians...*] /A1”

S-B: “[*Hanssen*]/A0 sold [*FBI secrets*]/A1 to [*the Russians*]/A3”

T-A: “[*Hanssen*]/A0 sold [*FBI secrets*]/A1 to [*the Russians*]/A3”

During preprocessing, the system does not detect an embedding of “sold” in “reported”, and so does not attach the attribute and value “CONFIDENCE: REPORTED” to the verb “sold” in S. During the subsumption check, the “sold” verbs now match, as both are considered factual.

### Qualifier Module

The Qualifier module allows comparison of qualifiers such as “all”, “some”, “any”, “no”, etc. In the experimental results summarized below, adding the Qualifier module does not improve performance, because in all the PARC examples involving qualifiers, a different qualifier in S and T corresponds to a negative label.

However, the qualifier module will correctly analyze the following sentence pair, which is a reordered pair from the PARC corpus:

S: *All soldiers were killed in the ambush.*

T: *Many soldiers were killed in the ambush.*

The default rule – non-identical argument modifiers cause subsumption to fail – is incorrect here, as S entails T. The Qualifier module correctly identifies the entailment of “many” by “all”, and subsumption will succeed.

## 5.1 Experimental Results

We present the evaluation results across two dimensions. On the horizontal axis we represent the baseline, system version#1 (SRL+LLM), and system version#2 (SRL + Deep structure). On the vertical axis we represent various knowledge modules used to enrich the basic representation. The results are summarized below in two tables. Table 1 represents the evaluation of the system with and without the KB inference rules and when SRL is perfect (i.e., considering only examples on which our SRL tool gives correct annotation). Table 2 shows the performance when the SRL system is used with the entire dataset, including those examples on which the SRL tool makes mistakes.

The results obtained with perfect semantic argument structure (perfect SRL) are provided here to illustrate the advantages of the hierarchical approach, as noise introduced by SRL errors can obscure the effects of the different levels of the hierarchical representation/subsumption.

The improvement across the vertical dimension, which represents additional analysis resources that use the structural information supported by the given system configuration, is

monotonic, indicating the benefits of semantic analysis of the structural information.

The improvement across the horizontal dimension, which represents successively finer structural representation, is clearly best for the system with full parse information. To summarize, the system behaves consistently, showing improvement as additional hierarchical structure and additional semantic analysis resources are added. These results show that the hierarchical approach is valid.

## Work In Progress

Presently, the system works well when the SRL annotation it depends on is mostly or completely correct. We are working on the following items to further improve the system’s performance:

- **Back-off Strategies to Handle Missing SRL Information.** We are working on a secondary step in each of the first two levels of the subsumption algorithm, in which we will use the Full Parse/dependency information to seek candidates to substitute for “missing” arguments and even verbs.
- **Verb Phrase Matching.** While the present SRL annotation allows us to handle simple verb phrases by decomposition, this is potentially error-prone. We are developing a module to address the problem of verb phrases (as highlighted in example 2 above) by preprocessing sentences to collapse simple verb phrases into a single node. This requires identifying the main verb to associate with the replacement node, and adding attributes to represent any qualification associated with the supporting verb (in this case, “manage” does not require new attributes at our current level of operation, but phrases like “failed to enter” and “tried to enter” would require negation and intention attributes respectively, both of which should result in subsumption failing unless similar qualifications are present in the matching verb/verb phrase).
- **More sophisticated quantity matching.** Clearly there is a need to use numbers as a critical matching element in the phrase-level subsumption phase. However, matching numbers is not necessarily straightforward, as they can be qualified in a number of ways – “at least \$1 million”, “over \$1 million”, “more than \$1 million”, “one million dollars”, etc. We can identify number boundaries in many cases with our Named Entity Tagger, and use this to identify number qualifiers of key entities. However, determining whether numbers match requires further processing to relate qualifiers, different notations, etc.

## 6 Previous Work

Knowledge representation and reasoning techniques have been studied in NLP for a long time [Schubert, 1986; Moore, 1986; Hobbs *et al.*, 1988]. Most approaches relied on First Order Logic representations with a general prover and without using acquired rich knowledge sources.

	without KB			with KB		
	LLM	SRL+LLM	SRL + Deep structure	LLM	SRL+LLM	SRL + Deep structure
<b>Base</b>	59.38	56.25	62.50	62.50	60.94	68.75
<b>VP</b>	N/A	57.81	62.50	N/A	67.19	75.00
<b>DA</b>	N/A	N/A	71.88	N/A	N/A	82.81
<b>Qual</b>	N/A	N/A	71.88	N/A	N/A	82.81

Table 1: System’s performance obtained for the PARC question-answer pairs without with perfect SRL. This corresponds to 64 question-answering pairs. The empty buckets (N/A) indicate that the module in the left hand column could not be used with that column’s system configuration.

	without KB			with KB		
	LLM	SRL+LLM	SRL + Deep structure	LLM	SRL+LLM	SRL + Deep structure
<b>Base</b>	61.84	55.26	61.84	64.47	59.21	67.11
<b>VP</b>	N/A	55.26	60.52	N/A	63.16	69.74
<b>DA</b>	N/A	N/A	68.42	N/A	N/A	77.63
<b>Qual</b>	N/A	N/A	68.42	N/A	N/A	77.63

Table 2: System’s performance obtained for the PARC question-answer pairs on the full data set. The empty buckets (N/A) show that no knowledge information could be used.

Significant development in NLP, specifically the ability to acquire knowledge and induce some level of abstract representation could, in principle, support more sophisticated and robust approaches. Nevertheless, most modern approaches developed so far are based on shallow representations of the text that capture lexico-syntactic relations based on dependency structures and are mostly built from grammatical functions in an extension to keyword-base matching [Durme *et al.*, 2003]. Some systems make use of some semantic information, such as WordNet lexical chains [Moldovan *et al.*, 2003], to slightly enrich the representation. Other have tried to learn various logic representations [Thompson *et al.*, 1997]. However, none of these approaches makes global use of a large number of resources as we do, or attempts to develop a flexible, hierarchical representation and an inference algorithm for it, as we present here.

## 7 Conclusions

This paper presents a principled, integrated approach to *semantic entailment*. We developed an expressive knowledge representation that provides a hierarchical encoding of structural, relational and semantic properties of the text and populated it using a variety of machine learning based tools. An inferential mechanism over a knowledge representation that supports both abstractions and several levels of representations allows us to begin to address important issues in abstracting over the variability in natural language. Our preliminary evaluation is very encouraging, yet leaves a lot to hope for. Improving our resources and developing ways to augment the KB are some of the important steps we need to take. Beyond that, we intend to tune the inference algorithm by incorporating a better mechanism for choosing the appropriate level at which to require subsumption. Given the fact that we optimize a linear function, it is straightforward to learn the cost function. Moreover, this can be done in such a way that the decision list structure is maintained.

## 8 Acknowledgments

We are grateful to Ron Kaplan and Xerox PARC for sharing the collection of question-answer pairs with us. This research is supported

by the Advanced Research and Development Activity (ARDA)’s Advanced Question Answering for Intelligence (AQUAINT) Program, a DOI grant under the Reflex program, NSF grants ITR-IIS-0085836, ITR-IIS-0085980, and an ONR MURI Award.

## References

- [Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *Description Logic Handbook*. Cambridge, 2003.
- [Braz *et al.*, ] R. Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. An inference model for semantic entailment in natural language. In *Proceedings of the National Conference on Artificial Intelligence*.
- [Collins, 1999] M. Collins. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, Computer Science Department, University of Pennsylvania, Philadelphia, 1999.
- [Cumby and Roth, 2002] C. M. Cumby and D. Roth. Learning with feature description logics. In S. Matwin and C. Sammut, editors, *The 12th International Conference on Inductive Logic Programming (ILP-02)*, pages 32–47. Springer, 2002. LNAI 2583.
- [Dagan and Glickman, 2004] I. Dagan and O. Glickman. Probabilistic textual entailment: Generic applied modeling of language variability. In *Learning Methods for Text Understanding and Mining*, Grenoble, France, 2004.
- [Durme *et al.*, 2003] B. Van Durme, Y. Huang, A. Kupsc, and E. Nyberg. Towards light semantic processing for question answering. HLT Workshop on Text Meaning, 2003.
- [Even-Zohar and Roth, 2001] Y. Even-Zohar and D. Roth. A sequential model for multi class classification. pages 10–19, 2001.
- [Fellbaum, 1998] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [Hobbs *et al.*, 1988] J. R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *Proc. of the 26th ACL*, pages 95–103, 1988.

- [Kingsbury *et al.*, 2002] P. Kingsbury, M. Palmer, and M. Marcus. Adding semantic annotation to the Penn treebank. In *Proceedings of the Human Language Technology conference (HLT)*, San Diego, CA, 2002.
- [Li *et al.*, 2004] X. Li, P. Morie, and D. Roth. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *Proceedings of the National Conference on Artificial Intelligence*, 2004.
- [Lin and Pantel, 2001] D. Lin and P. Pantel. DIRT: discovery of inference rules from text. In *KDD '01*, pages 323–328, 2001.
- [Lloyd, 1987] J. W. Lloyd. *Foundations of Logic Programming*. Springer, 1987.
- [Moldovan *et al.*, 2003] D. Moldovan, C. Clark, S. Harabagiu, and S. Maiorano. Cogex: A logic prover for question answering. In *HLT-NAACL*, 2003.
- [Moore, 1986] R. C. Moore. Problems in logical form. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*. Kaufmann, Los Altos, CA, 1986.
- [Punyakanok *et al.*, 2004] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proc. of the 20th International Conference on Computational Linguistics (COLING)*, Geneva, Switzerland, August 2004.
- [Punyakanok *et al.*, 2005] V. Punyakanok, D. Roth, and W. Yih. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.
- [Schubert, 1986] L. K. Schubert. From english to logic: Context-free computation of 'conventional' logical translations. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*. Kaufmann, Los Altos, CA, 1986.
- [Thompson *et al.*, 1997] C. Thompson, R. Mooney, and L. Tang. Learning to parse NL database queries into logical form. In *Workshop on Automata Induction, Grammatical Inference and Language Acquisition*, 1997.
- [Xpress-MP, ] Xpress-MP. Dash Optimization. Xpress-MP. <http://www.dashoptimization.com/products.html>.