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

Abstract

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. This paper presents a principled approach to this problem that builds on inducing representations of text snippets into a hierarchical knowledge representation along with a sound optimization-based inferential mechanism that makes use of it to decide semantic entailment. A preliminary evaluation on the PASCAL text collection is presented.

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 "*WalMart 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 & 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).

Several decades of research in natural language processing and related fields have made clear that the use of deep structural, relational and semantic properties of text is a necessary step towards supporting higher level tasks. 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. Just resorting to general purpose knowledge representations – FOL, probabilistic or hybrids – along with their corresponding general purpose inference algorithms, requires the ability to map surface representations into these languages, and may lead to brittleness and complexity problems.

We describe an integrated approach that provides solutions to the challenges mentioned above. 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 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.

We present a principled computational approach to *semantic entailment* in natural language that addresses some of the key problems encountered in traditional approaches – knowledge acquisition and brittleness. The solution includes a hierarchical knowledge representation language into which we induce appropriate representations of the given text and required background knowledge. The other main element is a sound inferential mechanism that makes use of the induced representation to determine an extended notion of subsumption, using an optimization approach that supports abstracting over language variability and representation inaccuracies. Along with describing the key elements of our approach, we present a preliminary system that implements it, and a preliminary evaluation of this system.

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 the following components:
KR: A Description Logic based hierarchical knowledge representation, into which we 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. Each rule is written as $lhs \sqsubseteq rhs$ describing subsumption relation between two representations, lhs (body of the rule) and rhs (the head of the rule).
Subsumption: An extended subsumption algorithm which determines subsumption between two representations. "Extended" here means that the basic unification operator is extended to support several word level and phrase level abstractions.

Our process starts with a set of machine learning based resources 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 surface representation of the source text S by augmenting it with heads of subsumed re-write 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 new 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.

This paper focuses on the inference algorithm, mostly in the second stage, and leaves many of the other details to a companion paper (Braz *et al.* 2005), for obvious space constraints. Along with the formal definition and justification developed here for our computational approach to semantic entailment, our knowledge representation and algorithmic approach provide a novel (preliminary) 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.

Algorithmic Semantic Entailment

Let $\mathcal{R}$ be a knowledge representation language with a well defined syntax and semantics over any 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 representations of two text snippets S, T, under this mapping by r_S, r_T, respectively. Note that we will use the word *expression* and *representation* interchangeably. 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 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 $r \in R$ 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: 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

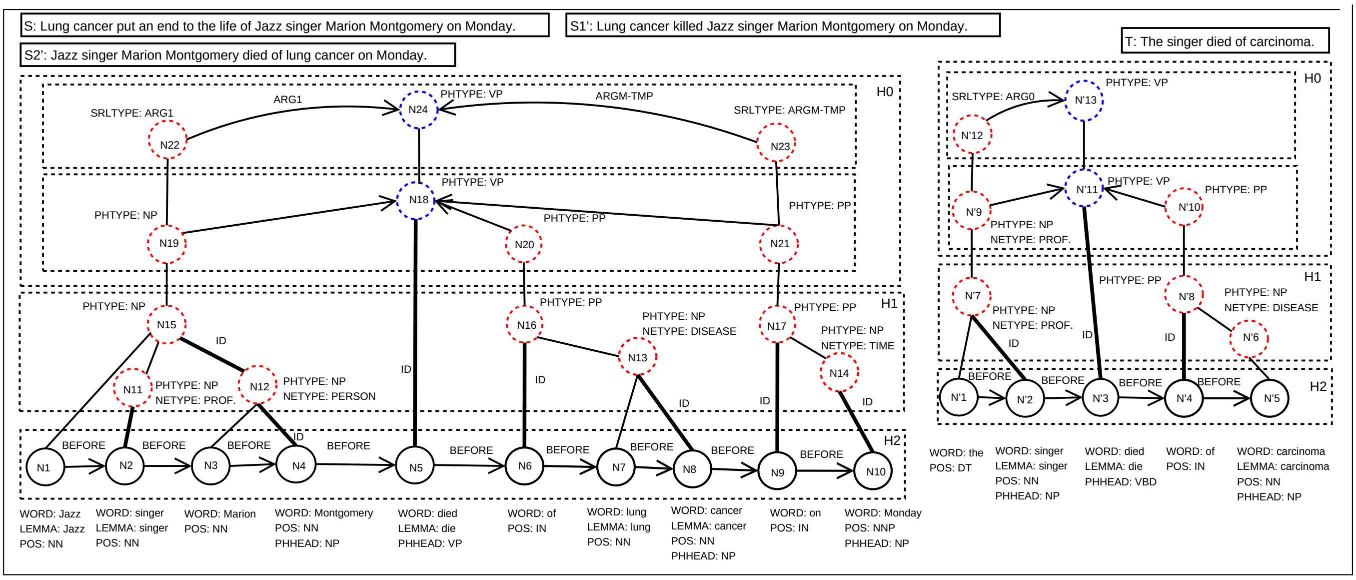

Figure 1: Example of *Source & Target* pairs represented as concept graphs. The original source sentence S generated several alternatives including S_1' and the sentence in the figure (S_2'). Our algorithm was not able to determine entailment of the first alternative (as it fails to match in the extended 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 → W of Z”; #2: “X put end to Y’s life → 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.

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 new representation that is in fact equivalent to r_S. Therefore, given a collection of rules $\{(lhs \sqsubseteq rhs)\}$ we can chain their applications, and incrementally generate faithful representations of S. Consequently, this algorithm is a sound algorithm[1] for semantic entailment according to Def. 1, but it is not complete. Its success depends on the size and quality of the rule set[2] 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 a representation for S and T is induced, the algorithm incrementally searches the 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 representation of S and generate a new (equivalent) representation of S. KB consists of syntactic and semantic 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

[1] Soundness depends on a “correct” induction of the representation of the text; we do not address this theoretically here.

[2] 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.

the surface level text.

(2) Representation S_i's are processed via the extended 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, and is discussed later in the paper.

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 FDL (Cumby & 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, PREDICATE_VALUE, PHTYPE, PHHEAD, NETYPE, ARGTYPE , NEG}. The first three are word level and the next three are phrase level. PREDICATE_VALUE is the string corresponding to a predicate, NETYPE is the named entity of a phrase, ARGTYPE is the type of semantic argument as defined in PropBank (Kingsbury, Palmer, & Marcus 2002) and NEG is a negation attribute of predicates.

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

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 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 a part-of-speech tagger (Even-Zohar & Roth 2001), a syntactic parser (Collins 1999), a semantic parser (Punyakanok *et al.* 2004; Punyakanok, Roth, & Yih 2005), a named entity recognizer [3], and a name coreference system (Li, Morie, & Roth 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 & 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 also supports the inference process in recovering from inaccuracies in lower level representations. Consider, for example, the following pair of sentences, 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 building*]/A1]/A1."[4]

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).

Inference Model and Algorithm

In this section we focus on how the extended subsumption process exploits the hierarchical knowledge representation and how we model inference as optimization.

Modeling Hierarchy & Unification Functions

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 $H = \{H_0, \ldots, H_j\}$ where a lower value of j indicates higher hierarchical level (more important nodes). This hierarchical representation is derived from

[3] Named entity recognizer from Cognitive Computation Group, http://l2r.cs.uiuc.edu/∼cogcomp

[4] The verbs "*manage*" and "*enter*" share the semantic argument "[*the bombers*]/A0".

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.$\forall(e, f) \in U_E$: their sinks and sources 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 that determines the cost of unifying pairs of nodes or 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 2 *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 λ_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, PREDICATE_VALUE, 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 (Kingsbury, Palmer, & Marcus 2002).

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.

`Hierarchy set` H_1 corresponds to phrase-level nodes and represents the semantic and syntactic arguments of the H_0 nodes (verbs). If the phase-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.

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

Inference as Optimization

We solve the subsumption problem by formulating an equivalent Integer Linear Programming (ILP) problem[5]. An ILP problem involves a set of integer variables $\{v_i\}$ and a set of linear equality or inequality constraints among them. Each variable v_i is associated with a cost c_i, and the problem is to find an assignment to the variables that satisfies the constraints and minimizes $\sum_i c_i v_i$.

To prove $S \sqsubseteq T$, we first start with the graph S (the initial graph). Then we extend S by adding the right hand sides of applicable rules. This is repeated up to a fixed number of rounds and results in an expanded graph S'_d. The formulation allows us to solve for the optimal unification from T to S'_d that minimizes the overall cost.

To formulate the problem this way, we need a set of variables that can represent different unifications from T to S'_d, and constraints to ensure the validity of the solution (ie, that the unification does not violate any nonnegotiable property). We explain this below. For readability, we sometimes express constraints in a logic form that can be easily transformed to linear constraints.

Representing Unification We introduce Boolean variables $u(n, m)$ for each pair of nodes $n \in V(T)$ and $m \in V(S'_d)$ in the same hierarchical level, and $u(e, f)$ for each pair of edges $e \in E(T)$ and $f \in E(S'_d)$ in the same level.

To ensure that the assignment to the matching variables represents a valid unification from T and S'_d, we need two types of constraints. First, we ensure the unification preserves the node and edge structure. For each pair of edges $e \in E(T)$ and $f \in E(S'_d)$, let n_i, n_j, m_k, and m_l be the sources and the sinks of e and f respectively. Then $u(e, f) \Rightarrow u(n_i, m_k) \bigwedge u(n_j, m_l)$. Finally, to ensure that the unification is a 1-to-1 mapping from T to

[5]Despite the fact that this optimization problem is NP hard, commercial packages have very good performance on sparse problems such as Xpress-MP by Dash Optimization, http://www.dashoptimization.com .

S'_d, $\forall n_i \in V(T) \sum_{m_j \in S'_d} u(n_i, m_j) = 1$, and $\forall m_j \in V(S'_d) \sum_{n_i \in T} u(n_i, m_j) \leq 1$.

Finding A Minimal Cost Solution We seek the unification with a minimum (and, of course, finite) cost: $\sum_{H_j} \sum_{u(x,y)|x,y \in H_j} \lambda_j G(x,y)u(x,y)$, where λ_j is the constant and G the cost of unification as we explained in the previous sections. The minimal subgraph S'_i of S'_d that T is unified to is also the minimal representation of S that incurs minimal unification cost.

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 mapping to First Order Logic representations with a general prover and without using acquired rich knowledge sources.

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, Mooney, & Tang 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.

Experimental Evaluation

We tested our approach on the PASCAL challenge data set [6]. As the system was designed to test for semantic entailment, the PASCAL data set is ideally suited, being composed of 276 source - target sentence pairs, with a truth value indicating whether the source logically entails the target. The set is split into various tasks: CD (Comparable Documents), IE (Information Extraction), IR (Information Retrieval), MT (Machine Translation), PP (Prepositional Paraphrases), QA (Question Answering), and RC (Reading Comprehension). The average sentence size varies from 11 (IR task) to 25 (MT task) words. Table 1 shows the system's performance.

As a baseline we use a lexical-level matching based on bag-of-words representation with lemmatization and normalization (LLM), which is a non-trivial baseline used in many information retrieval tasks.

Our system does particularly well (84.00% and 87.5%) on the QA and MT subtasks with a good corpus coverage of about 30%, reducing overall error by 68% compared to the LLM system. On average, our system offers a significant improvement over LLM, reducing overall error by over 20% across all categories. It took about 50 minutes to run the system on the corpus considered.

[6] http://www.pascal-network.org/Challenges/RTE/

	Overall [%]	Task [%]						
		CD	IE	IR	MT	PP	QA	RC
System	65.9	74.0	50.0	62.0	87.5	63.8	84.0	52.9
LLM	54.7	64.0	50.0	50.0	75.0	55.2	50.0	52.9

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

The following examples show some of the strengths and weaknesses of the system at different levels of the representation. They are intended to highlight the subsumption process at different levels of our hierarchical representation and not intended to represent the most complex sentences the system can handle.

Example 1

S1: "*The bombers had not managed to enter the building.*"

T1: "*The bombers entered the building.*"

Our system identifies two verb frames in S:

S1-A: "*[The bombers]/A0 [not]/AM_NEG manage [to enter the building]/A1*"

S1-B: "*[The bombers]/A0 [not]/AM_NEG enter [the building]/A1*"

and one verb frame for T:

T1-A: "*[The bombers]/A0 enter [the building]/A1.*"

The subsumption algorithm attempts to match T1-A's "enter" with both S1-A's "manage" and S1-B's "enter": there is no match between T1-A's "enter" and S1-A's "manage"; the match between T1-A's "enter" and S1-B's "enter" fails because the system has identified a negation attached to the verb "enter" in S1-B, and finds none attached to its counterpart in T1-A. Thus the system correctly determines that S does not entail T at the verb level.

The following example highlights the importance of Functional Subsumption.

Example 2

S2: "*The Spanish leader razed Tenochtitlan in 1521 and constructed a Spanish city on its ruins.*"

T2: "*The Spanish leader destroyed Tenochtitlan and built a Spanish city in its place.*"

Our system identifies two verb frames in both S and T:

S2-A: "*[The Spanish leader]/A0 raze [Tenochtitlan]/A1*"

S2-B: "*[The Spanish leader]/A0 construct [a Spanish city]/A1 [on its ruins]/AM_LOC*"

T2-A: "*[The Spanish leader]/A0 destroy [Tenochtitlan]/A1*"

T2-B: "*[The Spanish leader]/A0 build [a Spanish city]/A1 [in its place]/AM_LOC*"

In this case, the lemmas of the key verbs in S and T will not match without a successful Functional Subsumption call. Since WordNet contains synonymy relations for "destroy" and "raze", and "build" and "construct", the functional subsumption call determines that the verbs match. Consequently, the subsumption algorithm determines that, at the verb level, S entails T.

Example 3

S3: “*A car bomb that exploded outside a U.S. military base near Beiji killed 11 Iraqi citizens.*”

T3: “*A car bomb exploded outside a U.S. base in the northern town of Beiji killing 11 civilians.*”

The verb frames found by our system are:

S3-A: “*[A car bomb]/A1 explode [outside a U.S. military base near Beiji]/AM_LOC*”

S3-B: “*[A car bomb]/A0 kill [11 Iraqi citizens]/A1*”

T3-A: “*[A car bomb]/A1 explode [near Beiji]/AM_LOC*”

T3-B: “*[A car bomb]/A1 kill [11 civilians]/A1*”

Our system uses WordNet to relate “civilians” to “citizens”, allowing the corresponding A1 phrases to match; all other argument phrases in T in this case are simple substrings of their counterparts in S, and so the system correctly determines entailment of T by S.

Presently, our phrase-level matching algorithm does not give special weight to numbers; this can result in false positives in cases like the following:

Example 4

S4: “*Jennifer Hawkins is the 21-year-old beauty queen from Australia.*”

T4: “*Jennifer Hawkins is Australia’s 20-year-old beauty queen.*”

S4-A: “*[Jennifer Hawkins]/A1 is [the 21-year-old beauty queen from Australia]/A2*”

T4-A: “*[Jennifer Hawkins]/A1 is [Australia’s 20-year-old beauty queen]/A2*”

Our system matches almost all the key words in T4-A’s A2 with those in S4-A’s A2; as numbers do not yet carry more weight than other word elements, our system allows subsumption, resulting in a false positive.

Conclusions and Future Work

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 straight forward to learn the cost function. Moreover, this can be done in such a way that the decision list structure is maintained.

Acknowledgments

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 ONRs TRECC and NCASSR programs.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. 2003. *Description Logic Handbook*. Cambridge.

Braz, R.; Girju, R.; Punyakanok, V.; Roth, D.; and Sammons, M. 2005. Knowledge representation for semantic entailment and question-answering. In *IJCAI’05 Workshop on Knowledge and Reasoning for Answering Questions*.

Collins, M. 1999. *Head-driven Statistical Models for Natural Language Parsing.* Ph.D. Dissertation, Computer Science Department, University of Pennsylvenia, Philadelphia.

Cumby, C. M., and Roth, D. 2003. Learning with feature description logics. In Matwin, S., and Sammut, C., eds., *The 12th International Conference on Inductive Logic Programming (ILP)*. Springer. LNAI 2583.

Dagan, I., and Glickman, O. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. In *Learning Methods for Text Understanding and Mining*.

Durme, B. V.; Huang, Y.; Kupsc, A.; and Nyberg, E. 2003. Towards light semantic processing for question answering. HLT Workshop on Text Meaning.

Even-Zohar, Y., and Roth, D. 2001. A sequential model for multi class classification. In *Proc. of the 2001 Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 10–19.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Hobbs, J. R.; Stickel, M.; Martin, P.; and Edwards, D. 1988. Interpretation as abduction. In *Proc. of the 26th Annual Meeting of the Association for Computational Linguistics (ACL)*, 95–103.

Kingsbury, P.; Palmer, M.; and Marcus, M. 2002. Adding semantic annotation to the Penn treebank. In *Proc. of the 2002 Human Language Technology conference (HLT).*

Li, X.; Morie, P.; and Roth, D. 2004. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *Proc. of the 19th National Conference on Artificial Intelligence (AAAI)*.

Lin, D., and Pantel, P. 2001. DIRT: discovery of inference rules from text. In *Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001*, 323–328.

Lloyd, J. W. 1987. *Foundations of Logic Progamming*. Springer.

Moldovan, D.; Clark, C.; Harabagiu, S.; and Maiorano, S. 2003. Cogex: A logic prover for question answering. In *Proc. of HLT-NAACL 2003*.

Moore, R. C. 1986. Problems in logical form. In Grosz, B. J.; Sparck Jones, K.; and Webber, B. L., eds., *Natural Language Processing*. Los Altos, CA: Kaufmann.

Punyakanok, V.; Roth, D.; Yih, W.; and Zimak, D. 2004. Semantic role labeling via integer linear programming inference. In *Proc. of the 20th International Conference on Computational Linguistics (COLING)*.

Punyakanok, V.; Roth, D.; and Yih, W. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*.

Schubert, L. K. 1986. From english to logic: Contex-free computation of ’conventional’ logical translations. In Grosz, B. J.; Sparck Jones, K.; and Webber, B. L., eds., *Natural Language Processing*. Los Altos, CA: Kaufmann.

Thompson, C.; Mooney, R.; and Tang, L. 1997. Learning to parse NL database queries into logical form. In *Workshop on Automata Induction, Grammatical Inference and Language Acquisition*.