Journal of Artificial Intelligence Research 54 (2015) 1-57 Submitted 09/14; published 09/15

Knowledge-Based Textual Inference via Parse-Tree Transformations

Roy Bar-Haim BARHAIR@GMAIL.COM

Ido Dagan DAGAN@CS.BIU.AC.IL
Computer Science Department, Bar-Ilan University
Ramat-Gan 52900, Israel

Jonathan Berant YONATAN@CS.STANFORD.EDU
Computer Science Department, Stanford University

Abstract

Textual inference is an important component in many applications for understanding natural language. Classical approaches to textual inference rely on logical representations for meaning, which may be regarded as "external" to the natural language itself. However, practical applications usually adopt shallower lexical or lexical-syntactic representations, which correspond closely to language structure. In many cases, such approaches lack a principled meaning representation and inference framework. We describe an inference formalism that operates directly on language-based structures, particularly syntactic parse trees. New trees are generated by applying inference rules, which provide a unified representation for varying types of inferences. We use manual and automatic methods to generate these rules, which cover generic linguistic structures as well as specific lexical-based inferences. We also present a novel packed data-structure and a corresponding inference algorithm that allows efficient implementation of this formalism. We proved the correctness of the new algorithm and established its efficiency analytically and empirically. The utility of our approach was illustrated on two tasks: unsupervised relation extraction from a large corpus, and the Recognizing Textual Entailment (RTE) benchmarks.

1. Introduction

Textual inference in Natural Language Processing (NLP) is concerned with deriving target meanings from texts. In the *textual entailment* framework (Dagan, Roth, Sammons, & Zanzotto, 2013), this is reduced to inferring a textual statement (the *hypothesis* h) from a source *text* (t). Traditional approaches in formal semantics perform such inferences over logical forms derived from the text. By contrast, most practical NLP applications avoid the complexities of logical interpretation. Instead, they operate over shallower representations such as parse trees, possibly supplemented with limited semantic information about named entities, semantic roles, and so forth. This was clearly demonstrated in the recent PASCAL Recognizing Textual Entailment (RTE) Challenges (Dagan, Glickman, & Magnini, 2006b; Bar-Haim, Dagan, Dolan, Ferro, Giampiccolo, Magnini, & Szpektor, 2006; Giampiccolo, Magnini, Dagan, & Dolan, 2007; Giampiccolo, Trang Dang, Magnini, Dagan, & Dolan, 2008; Bentivogli, Dagan, Dang, Giampiccolo, & Magnini, 2009; Bentivogli, Clark, Dagan,

Dang, & Giampiccolo, 2010), a popular framework for evaluating application-independent semantic inference.[1]

Inference over such representations is commonly made by applying transformations or substitutions to the tree or graph representing the text. These transformations are based on available knowledge about paraphrases, lexical relations such as synonyms and hyponyms, syntactic variations, and more (de Salvo Braz, Girju, Punyakanok, Roth, & Sammons, 2005; Haghighi, Ng, & Manning, 2005; Kouylekov & Magnini, 2005; Harmeling, 2009). Such transformations may be generally viewed as *inference rules*. Some of the available semantic knowledge bases were composed manually, either by experts, for example WordNet (Fellbaum, 1998), or by a large community of contributors, such as the Wikipedia-based DBPedia resource (Lehmann et al., 2009). Other knowledge bases were learned automatically through distributional and pattern-based methods, or by using aligned monolingual or bilingual parallel texts (Lin & Pantel, 2001; Shinyama, Sekine, Sudo, & Grishman, 2002; Szpektor, Tanev, Dagan, & Coppola, 2004; Chklovski & Pantel, 2004; Bhagat & Ravichandran, 2008; Ganitkevitch, Van Durme, & Callison-Burch, 2013). Overall, applied knowledge-based inference is a prominent line of research that has gained much interest. Recent examples include the series of workshops on *Knowledge and Reasoning for Answering Questions* (Saint-Dizier & Mehta-Melkar, 2011) and the evaluation of knowledge resources in the recent Recognizing Textual Entailment challenges (Bentivogli et al., 2010).

While many applied systems use semantic knowledge through such inference rules, their use is typically limited, application-specific, and somewhat heuristic. Formalizing these practices is important for textual inference research, analogous to the role of well-formalized models in parsing and machine translation. We take a step in this direction by introducing a generic inference formalism over parse trees. Our formalism uses inference rules to capture a wide variety of inference knowledge in a simple and uniform manner, and specifies a small set of operations that suffice to broadly utilize such knowledge.

In our formalism, applying an inference rule has a clear, intuitive interpretation of generating a new sentence parse (a *consequent*), semantically entailed by the source sentence. The inferred consequent may be subject to further rule applications, and so on. Rule applications may be independent of each other, modifying disjoint parts of the source tree, or may specify mutually-exclusive alternatives (e.g., different synonyms for the same source word). Deriving the hypothesis from the text is analogous to proof search in logic, where the propositions are parse trees and deduction steps correspond to rule applications.

A naïve implementation of the formalism would generate each consequent explicitly as a separate tree. However, as we discuss further in Section 5, such implementation raises severe efficiency issues, since the number of consequents may grow exponentially in the number of possible rule applications. Previous work proposed only partial solutions to this problem (cf. Section 8). In this work we present a novel data-structure, termed *compact forest*, for packed representation of entailed consequents, and a corresponding inference algorithm. We prove that the new algorithm is a valid implementation of the formalism, and establish its efficiency both analytically, showing typical exponential-to-linear reduction, and empirically, showing improvement by orders of magnitude. Together, our formalism and

1. See, for instance, the listing of techniques per submission that was provided by the organizers of the first three challenges (Dagan et al., 2006b; Bar-Haim et al., 2006; Giampiccolo et al., 2007).

its novel efficient inference algorithm open the way for large-scale rule application within a well-formalized framework.

Based on our formalism and inference algorithm, we built an inference engine that incorporates a variety of semantic and syntactic knowledge bases (cf. Section 6). We evaluated the inference engine on the following tasks:

1. Unsupervised relation extraction from a large corpus. This setting allows evaluation of knowledge-based inferences over a real-world distribution of texts.

2. Recognizing textual entailment (RTE). To cope with the more complex RTE examples, we complemented our knowledge-based inference engine with a machine-learning-based entailment classifier, which provides necessary approximate matching capabilities.

The inference engine was shown to have a substantial contribution to both tasks, illustrating the utility of our approach.

Bar-Haim, Dagan, Greental, and Shnarch (2007) and Bar-Haim, Berant, and Dagan (2009) described earlier versions of the inference framework and the algorithm for its efficient implementation, respectively. The current article includes major enhancements to both of these contributions. The formalism is presented in more detail, including further examples and pseudo-code for its algorithms. We present several extensions to the formalism, including treatment of co-reference, traces and long-range dependencies, and enhanced modeling of polarity. The efficient inference algorithm is also presented in more detail, including its pseudo-code. In addition, we provide complete proofs for the theorems, which establish the correctness of this algorithm. Finally, the article contains an extended analysis of the inference component in our RTE system, in terms of applicability, coverage, and the correctness of rule applications.

2. Background

In this section, we provide background on textual entailment. We then survey the various approaches applied to the task of Recognizing Textual Entailment (RTE). In particular, we focus on the use of semantic knowledge within current RTE systems.

2.1 Textual Entailment

Many semantic applications need to identify that the same meaning is expressed by, or can be inferred from, various language expressions. For example, Question-Answering systems need to verify that the retrieved passage text entails the selected answer. Given the question *"Who is John Lennon's widow?"*, the text *"Yoko Ono unveiled a bronze statue of her late husband, John Lennon, to complete the official renaming of England's Liverpool Airport as Liverpool John Lennon Airport."* entails the expected answer *"Yoko Ono is John Lennon's widow"*[2]. Similarly, Information Extraction systems need to validate that the given text indeed entails the semantic relation that is expected to hold between the extracted slot fillers (e.g., *'X works for Y'*). Information Retrieval queries such as "Alzheimer's drug treatment"[3]

2. The example is taken from the RTE-2 dataset (Bar-Haim et al., 2006).
3. This was one of the topics in the TREC-6 IR benchmark (Voorhees & Harman, 1997).

can be rephrased as propositions (e.g., "Alzheimer's disease is treated using drugs"), which are expected to be entailed from relevant documents. When selecting sentences to be included in the summary, multi-document summarization systems should verify that the meaning of the candidate sentence is not entailed by sentences already in the summary, to avoid redundancy.

This observation led Dagan and Glickman to propose a unifying framework for modeling language variability, termed *Textual Entailment* (TE) (Dagan & Glickman, 2004). Dagan et al. (2006b) define TE as follows:

> "We say that t entails h if, typically, a human reading t would infer that h is most likely true. This somewhat informal definition is based on (and assumes) common human understanding of language as well as common background knowledge."

Dagan et al. (2013) further discuss TE definition and its relation to classical semantic entailment in linguistics literature. The Recognizing Textual Entailment Challenges (RTE), which have been held annually since 2004 (Dagan et al., 2006b; Bar-Haim et al., 2006; Giampiccolo et al., 2007, 2008; Bentivogli et al., 2009, 2010), have formed a growing research community around this task.

The holy grail of TE research is the development of *entailment engines*, to be used as generic modules within different semantic applications, similar to the current use of syntactic parsers and morphological analyzers. Since textual entailment is defined as a relation between surface texts, it is not bound to a particular semantic representation. This allows a "black-box" view of the entailment engine, where the input/output interface is independent from the internal implementation, which may employ different types of semantic representations and inference methods.

2.2 Determining Entailment

Consider the following *(t,h)* pair[4]:

t	The oddest thing about the UAE is that only 500,000 of the 2 million people living in the country are UAE citizens.
h	The population of the United Arab Emirates is 2 million.

Understanding that $t \Rightarrow h$ involves several inference steps. First, we infer from the reduced relative clause in *"2 million people living in the country"* the proposition:

(1) *2 million people live in the country.*

Next, we observe that *"the country"* refers to *"the UAE"*, so we can rewrite (1) as

(2) *2 million people live in the UAE.*

Knowing that *"UAE"* is an acronym for *"United Arab Emirates"*, we further obtain:

(3) *2 million people live in the United Arab Emirates.*

4. Taken from the RTE1 test set (Dagan et al., 2006b).

We finally paraphrase this to obtain h:

(4) *The population of the United Arab Emirates is 2 million.*

In general, textual inference involves diverse linguistic and world knowledge, including knowledge of relevant syntactic phenomena (e.g., relative clauses), paraphrasing ('*X people live in Y*→ *the population of Y is X*'), lexical knowledge ('*UAE*→ *United Arab Emirates*'), and so on. It may also require co-reference resolution, for example, substituting *"the country"* with *"UAE"*. We may think of all these types of knowledge as representing *inference rules* that define derivation of new entailed propositions or *consequents*. In this work we introduce a formal inference framework based on inference rule application. For the current discussion, however, an informal notion of inference rules would suffice.

The above example illustrates the derivation of h from t through a sequence of inference rule applications, a procedure generally known as *forward chaining*. Finding the sequence of rule applications that would get us from t to h (or as close as possible) is thus a search problem, defined over the space of all possible rule application chains.

Ideally, we would like to base our entailment engine solely on trusted knowledge-based inferences. In practice, however, available knowledge is incomplete, and full derivation of h from t is often not feasible. Therefore, requiring strict knowledge-based "proofs" is likely to yield limited recall. Alternatively, we may back off to a more heuristic approximate entailment classification.

The next two sections survey these two complementary inference types: knowledge-based inference, which is the focus of this research, and approximate entailment matching and classification.

2.3 Knowledge-Based Inference

In this section, we describe some of the common resources for inference rules (2.3.1), and their use in textual entailment systems (2.3.2).

2.3.1 Semantic Knowledge Resources

Lexical Knowledge Lexical-semantic relations between words or phrases play an important role in textual inference. The most prominent lexical resource is WordNet (Fellbaum, 1998), a manually composed wide-coverage lexical-semantic database. The following WordNet relations are typically used for inference: *synonyms* ('*buy* ↔ *purchase*'), *antonyms* ('*win* ↔ *lose*'), *hypernyms/hyponyms* ("is-a" relations, '*violin*→ *musical instrument*'), *meronyms* ("part-of" relations, '*Provence*→ *France*') and *derivations* such as '*meeting*→ *meet*'.

Many researchers aimed at deriving lexical relations automatically, using diverse methods and sources. Much of this automatically-extracted knowledge is complementary to WordNet, however, it is typically less accurate. Snow, Jurafsky, and Ng (2006a) presented a method for automatically expanding WordNet with new synsets, achieving high precision. Lin's thesaurus (Lin, 1998) is based on distributional similarity. Recently, several works aimed to extract lexical-semantic knowledge from Wikipedia, using its metadata, as well as textual definitions (Kazama & Torisawa, 2007; Ponzetto & Strube, 2007; Shnarch, Barak, & Dagan, 2009; Lehmann et al., 2009, and others). For a recent empirical study on the

inferential utility of common lexical resources, see the work of Mirkin, Dagan, and Shnarch (2009).

Paraphrases and Lexical-Syntactic Inference Rules These rules typically represent entailment or equivalence between predicates, including the correct mapping between their arguments (e.g., '*acquisition of Y by X* → *X purchase Y*'). Much work has been dedicated to unsupervised learning of such relations from comparable corpora (Barzilay & McKeown, 2001; Barzilay & Lee, 2003; Pang, Knight, & Marcu, 2003), by querying the Web (Ravichandran & Hovy, 2002; Szpektor et al., 2004), or from a local corpus (Lin & Pantel, 2001; Glickman & Dagan, 2003; Bhagat & Ravichandran, 2008; Szpektor & Dagan, 2008; Yates & Etzioni, 2009). In particular, textual entailment systems have widely used the DIRT resource of Lin and Pantel. The common idea underlying these algorithms, is that predicates sharing the same argument instantiations are likely to be semantically related.

NomLex-Plus (Meyers, Reeves, Macleod, Szekeley, Zielinska, & Young, 2004) is a lexicon containing mostly nominalizations of verbs, with allowed argument structures (e.g., *'X's acquisition of Y'/'Y's acquisition by X'* etc.). *Argument-mapped WordNet (AmWN)* (Szpektor & Dagan, 2009) is a resource for inference rules between verbal and nominal predicates, including their argument mapping. It is based on WordNet and NomLex-Plus, and was verified statistically through intersection with the unary-DIRT algorithm (Szpektor & Dagan, 2008).

Syntactic Transformations Textual entailment often involves inference over generic syntactic phenomena such as passive/active transformations, appositions, conjunctions, etc., as illustrated in the following examples:

- John smiled and laughed ⇒ John laughed (conjunction)
- My neighbor, John, came in ⇒ John is my neighbor (apposition)
- The paper that I'm reading is interesting ⇒ I'm reading a paper (relative clause).

Syntactic transformations have been addressed to some extent by de Salvo Braz et al. (2005) and Romano, Kouylekov, Szpektor, Dagan, and Lavelli (2006). We describe a novel syntactic rule base for entailment, based on a survey of relevant linguistic literature, as well as on extensive data analysis (Sections 6.1–6.2).

2.3.2 The Use of Semantic Knowledge in Textual Entailment Systems

Following our description of common knowledge sources for textual inference, we now discuss the use of such knowledge in textual entailment systems.

Textual entailment systems usually represent t and h as trees or graphs, based on their syntactic parse, predicate-argument structure, and various semantic relations. Entailment is then determined by measuring how well h is *matched* (or *embedded*) in t, or by estimating the *distance* between t and h, commonly defined as the cost of transforming t into h. In the next section, we briefly cover various methods that have been proposed for approximate matching and heuristic transformations of graphs and trees. The role of semantic knowledge in this general scheme is to bridge the gaps between t and h that stem from language variability. For example, applying the lexical-semantic rule '*purchase*→ *buy*' to t allows the matching of the word *buy* appearing in h with the word *purchase* appearing in t.

Most RTE systems restrict both the type of allowed inference rules and the search space. Systems based on lexical (word-based or phrase-based) matching of h in t (Haghighi et al., 2005; MacCartney, Galley, & Manning, 2008) or on heuristic transformation of t into h (Kouylekov & Magnini, 2005; Harmeling, 2009) typically apply only lexical rules (without variables), where both sides of the rule are matched directly in t and h.

Hickl (2008) derived from a given (t, h) pair a small set of consequents that he terms *discourse commitments*. The commitments were generated by several different tools and techniques, based on syntax (conjunctions, appositions, relative clauses, etc.), co-reference, predicate-argument structure, the extraction of certain relations, and paraphrase acquisition from the Web. Pairs of commitments derived from t and h were fed into the next stages of the RTE system – lexical alignment and entailment classification. Prior to commitment generation, several linguistic preprocessing modules were applied to the text, including syntactic dependency parsing, semantic dependency parsing, named entity recognition, and co-reference resolution. Hickl employed a probabilistic finite-state transducer (FST)-based extraction framework for commitment generation, and extraction rules were modeled as a series of weighted regular expressions. The commitments in their textual form were fed back into the system, until no additional commitments were generated.

De Salvo Braz et al. (2005) were the first to incorporate syntactic and semantic inference rules in a comprehensive entailment system. In their system, inference rules are applied over hybrid syntactic-semantic structures called *concept graphs*. When the left hand side (LHS) of a rule is matched in the concept graph, the graph is augmented with an instantiation of the right hand side (RHS) of the rule. After several iterations of rule application, their system attempts to embed the hypothesis in the augmented graph. Other types of semantic knowledge, such as verb normalization and lexical substitutions, are applied either before rule application (at preprocessing time) or after rule application, as part of hypothesis subsumption (embedding).

Several entailment systems are based on logical inference. Bos and Markert (2005, 2006) represented t and h as DRS structures used in Discourse Representation Theory (Kamp & Reyle, 1993), which were then translated into first-order logic. Background knowledge (BK) was encoded as axioms, and comprised lexical relations from WordNet, geographical knowledge, and a small set of manually composed axioms encoding generic knowledge. Bos and Markert used a logic theorem prover to find a proof that t entails h (alone or together with the background knowledge BK), or that h and t are inconsistent with each other (implying non-entailment) or with the background knowledge. The logic prover was complemented by a model builder that aimed to find counter-examples (e.g., a model where $t \wedge \neg h$ holds). The logical inference system suffered from low coverage, due to the limited background knowledge available, and was able to find proofs only for a small fraction of the RTE2 dataset. Therefore, the RTE system of Bos and Markert combined logical inference with a shallow approximate matching method, based mainly on word overlap.

LCC's logic-based entailment system (Tatu & Moldovan, 2006) was one of the top performers in RTE2 and RTE3 (Tatu, Iles, Slavick, Novischi, & Moldovan, 2006; Tatu & Moldovan, 2007). It was based on proprietary tools for deriving rich semantic representations, and on extensive knowledge engineering. The syntactic parses of t and h were transformed into logic forms (Moldovan & Rus, 2001), and this representation was enriched with a variety of relations extracted by a semantic parser, as well as named entities and

temporal relations. Inference knowledge included on-demand axioms based on extended WordNet lexical chains, WordNet glosses, and NLP rewrite rules. Additional knowledge types included several hundreds of world knowledge axioms, temporal axioms, and semantic composition axioms (e.g., encoding the transitivity of the *kinship* relation). Based on the rich semantic representation and the extensive set of axioms, a theorem prover aimed to prove by refutation that t entails h. If the proof failed, h was repeatedly simplified until a proof was found, reducing the proof score with each simplification.

2.4 Approximate Entailment Classification

Semantic knowledge is always incomplete. Therefore, in most cases, knowledge-based inference must be complemented with approximate, heuristic methods for determining entailment. Most RTE systems employ only a limited amount of semantic knowledge, and focus on methods for approximate entailment classification. A common architecture for RTE systems (Hickl, Bensley, Williams, Roberts, Rink, & Shi, 2006; Snow, Vanderwende, & Menezes, 2006b; MacCartney, Grenager, de Marneffe, Cer, & Manning, 2006) comprises the following stages:

1. *Linguistic processing*: Includes syntactic (and possibly semantic) parsing, named-entity recognition, co-reference resolution, etc. Often, t and h are represented as trees or graphs, where nodes correspond to words and edges represent relations between words.

2. *Alignment*: Find the best mapping from h nodes to t nodes, taking into account both node and edge matching.

3. *Entailment classification*: Based on the alignment found, a set of features is extracted and passed to a classifier for determining entailment. These features measure the alignment quality, and also try to detect cues for false entailment. For example, if a node in h is negated but its aligned node in t is not negated, it may indicate false entailment.

An alternative approach aims to transform the text into the hypothesis, rather than aligning them. Kouylekov and Magnini (2005) applied a tree edit distance algorithm for textual entailment. Each edit operation (node insertion/deletion/substitution) is assigned a cost. The algorithm aims to find the minimum-cost sequence of operations that transform t into h. Mehdad and Magnini (2009b) proposed a method for estimating the cost of each edit operation based on Particle Swarm Optimization. Wang and Manning (2010) presented a probabilistic tree-edit approach that models edit operations using structured latent variables. Tree edits are represented as state transitions in a Finite-State Machine (FSM), and the model is parameterized as a Conditional Random Field (CRF). Harmeling (2009) developed a probabilistic transformation-based approach. He defined a fixed set of operations, including syntactic transformations, WordNet-based substitutions, and more heuristic transformations such as adding/removing a verb or a noun. The probability of each transformation was estimated from the development set. Similarly, Heilman and Smith (2010) classify entailment based on the sequence of edits transforming t to h. They employ more generic edit operations and a greedy search heuristic, which is guided by a cost function that measures the remaining distance from h using a tree kernel.

Zanzotto, Pennacchiotti, and Moschitti (2009) aimed to classify a given (t, h) pair by analogy to similar pairs in the training set. Their method is based on finding intra-pair alignment (i.e., between t and h) for capturing the transformation from t to h, and inter-pair alignment, capturing the analogy between the new pair (t, h) and a previously seen pair (t', h'). A cross-pair similarity kernel is then computed, based on tree kernel similarity applied to the aligned texts and the aligned hypotheses. Another cross-pair similarity kernel was proposed by Wang and Neumann (2007). They extracted *tree skeletons* from t and h, consisting of left and right *spines*, defined as unlexicalized paths starting at the root. They then found sections where t and h spines differ and compared these sections across pairs using a subsequence kernel.

3. Research Goal

The goal of textual entailment research is to develop entailment engines that can be used as generic inference components within various text-understanding applications. Logic-based entailment systems provide a formalized and expressive framework for textual inference. However, deriving logic representations from text is a complex task, and available tools do not match the accuracy and robustness of current syntactic parsers (which is often the basis for semantic parsing). Furthermore, interpretation into logic forms is often unnecessary, as many of the common inferences can be modeled with shallower representations.

It follows that most textual entailment systems (and text-understanding applications in general) operate over lexical-syntactic representations, possibly supplemented with some partial semantic annotation. However, unlike logic-based approaches, most of these systems lack a clear, unified formalism for knowledge representation and inference; instead they employ multiple representations and inference mechanisms. A notable exception is the natural logic framework of MacCartney and Manning (2009), which has a rather different focus than the current work. We discuss this further in Section 8.

In this work, we develop a well-formalized entailment approach for the lexical-syntactic level. Our formalism models a wide variety of inference rules and their composition, based on a unified representation and a small set of inference operations. Moreover, we present an efficient implementation of this formalism using a novel data structure and algorithm that allow compact representation of the proof search space.

We see the contribution of this work as both practical and theoretical. From a practical (or "engineering") perspective, our formalism may simplify the development of entailment systems, as the number of representations and inference mechanisms that need to be dealt with is minimal. Furthermore, our efficient implementation may allow entailment engines to explore much larger search spaces. From a theoretical perspective, concise, formal modeling leads to better insight into the phenomenon under investigation. In particular, having a formal model of an entailment engine makes it possible to apply formal methods for investigating its properties. This enabled us to prove the correctness of the efficient implementation of our formalism (cf. Appendix A). We next present our inference formalism.

Rule Type	Sources	Examples
Syntactic	Manually-composed	Passive/active, apposition, relative clause, conjunctions
Lexical - Syntactic	Learned with unsupervised algorithms (DIRT, TEASE), and derived automatically by integrating information from WordNet and Nomlex, verified using corpus statistics (AmWN)	*X's wife, Y → X is married to Y* *X bought Y → Y was sold to X* *X is a maker of Y → X produces Y*
Lexical	WordNet, Wikipedia	*steal→take*, *Albanian→Albania* *Janis Joplin→singer* *Amazon→South America*

Table 1: Representing diverse knowledge types as inference rules

4. An Inference Formalism over Parse Trees

The previous sections highlighted the need for a more principled, well-formalized approach for textual inference at the lexical-syntactic level. In this section, we propose a step towards filling this gap, by defining a formalism for textual inference over parse-based representations. All semantic knowledge required for inference is represented as *inference rules*, which encode parse tree transformations. Each rule application generates a new consequent sentence (represented as a parse tree) from a source tree. Figure 1b shows a sample inference rule, representing a passive-to-active transformation.

From a knowledge representation and usage perspective, inference rules provide a simple unifying formalism for representing and applying a very broad range of inference knowledge. Some examples of this breadth are illustrated in Table 1. From a knowledge acquisition perspective, representing inference rules at the lexical-syntactic level allows easy incorporation of rules learned by unsupervised methods, which is important for scaling inference systems. Interpretation into stipulated semantic representations, which is often difficult and is inherently a supervised semantic task for learning, is circumvented altogether. From a historical machine translation perspective, our approach is similar to transfer-based translation, as contrasted with semantic interpretation into Interlingua. Our overall research goal is to explore the reach of such an inference approach, and to identify the scope in which semantic interpretation may not be needed.

Given a syntactically parsed source text and a set of inference rules, our formalism defines the set of consequents derivable from the text using the rules. Each consequent is obtained through a sequence of rule applications, each generating an intermediate parse tree, similar to a proof process in logic. In addition, new consequents may be inferred based on co-reference relations and identified traces. Our formalism also includes *annotation rules* that add features to existing trees. According to the formalism, a text t *entails* a hypothesis h if h is a consequent of t.

In the rest of this section, we define and illustrate each of the formalism components: sentence representation (Section 4.1), inference rules and their application (Sections 4.2–4.3), inference based on co-reference relations and traces (Section 4.4), and annotation

Input: a source tree s ; a rule $E : L \rightarrow R$
Output: a set D of derived trees

$M \leftarrow$ the set of all matches of L in s
$D \leftarrow \emptyset$
for each $f \in M$ **do**
  $l \leftarrow$ the subtree matched by L in s according to match f

  // R instantiation
  $r \leftarrow$ a copy of R
  for each variable $v \in r$ **do**
    Instantiate v with $f(v)$
  for each aligned pair of nodes $u_L \in l$ and $u_R \in r$ **do**
    for each daughter m of u_L such that $m \notin l$ **do**
      Copy the subtree of s rooted in m under u_R in r, with the same dependency relation

  // Derived tree generation
  if *substitution* rule **then**
    $d \leftarrow s$ copy with l (and the descendants of its nodes) replaced by r
  else // *introduction* rule
    $d \leftarrow r$

  add d to D

Algorithm 1: Applying a rule to a tree

rules (Section 4.5). These components form an inference process that specifies the set of inferable consequents for a given text and a set of rules (Section 4.6). Section 4.7 extends the hypothesis definition, allowing h to be a template rather than a proposition. Finally, Section 4.8 discusses limitations and possible extensions of our formalism.

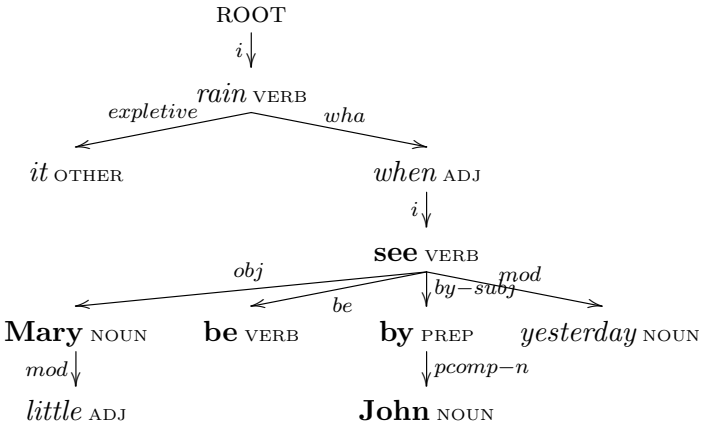
4.1 Sentence Representation

We assume that sentences are represented by some form of parse trees. In this work, we focus on dependency tree representation, which is often preferred to directly capture predicate-argument relations. Two dependency trees are shown in Figure 1a. Nodes represent words and hold a set of features and their values. These features include the word lemma and part-of-speech, and additional features that may be added during the inference process. Edges are annotated with dependency relations.

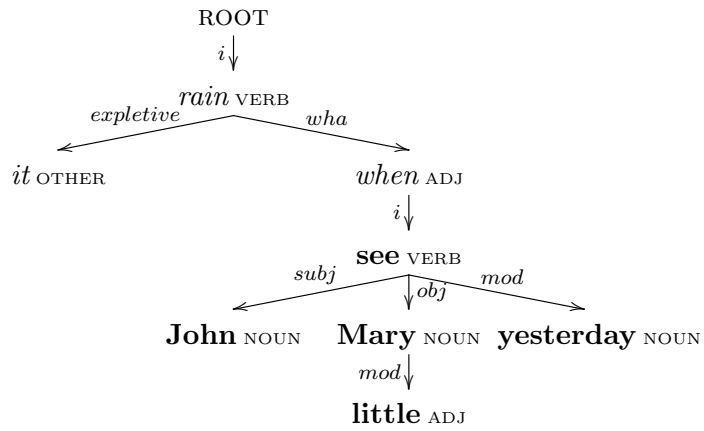
4.2 Inference Rules

An entailment (or *inference*) rule '$L \rightarrow R$' is primarily composed of two *templates*, *left-hand-side (LHS)* L and *right-hand-side (RHS)* R. Templates are dependency subtrees, which may contain POS-tagged *variables*, matching any lemma. Figure 1 shows a passive-to-active transformation rule, and illustrates its application.

The rule application procedure is given in Algorithm 1. Rule application generates a set D of derived trees (*consequents*) from a source tree s through the steps described below.

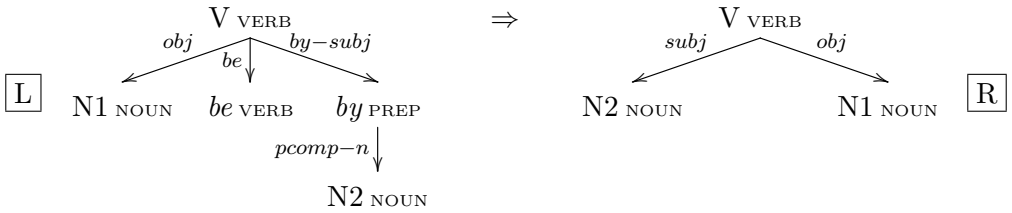


Source: It rained when little Mary was seen by John yesterday.



Derived: It rained when John saw little Mary yesterday.

(a) Passive-to-active tree transformation



(b) Passive to active substitution rule.

Figure 1: Application of an inference rule. POS and relation labels are based on Minipar (Lin, 1998). $N1$, $N2$ and V are variables, whose instances in L and R are implicitly aligned. The *by-subj* dependency relation indicates a passive sentence.

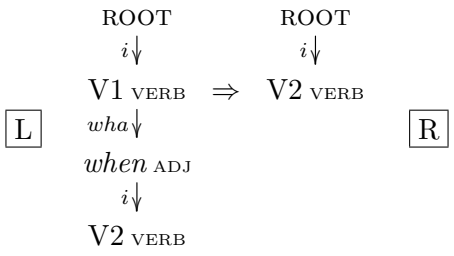


Figure 2: Temporal clausal modifier extraction (introduction rule)

4.2.1 L Matching

First, matches of L in the source tree s are sought. L is *matched* in s if there exists a one-to-one node mapping function f from L to s, such that:

1. For each node u in L, $f(u)$ has the same features and feature values as u. Variables match any lemma value in $f(u)$.

2. For each edge $u \rightarrow v$ in L, there is an edge $f(u) \rightarrow f(v)$ in s, with the same dependency relation.

If matching fails, the rule is not applicable to s. In our example, the variable V is matched in the verb *see*, $N1$ is matched in *Mary* and $N2$ is matched in *John*. If matching succeeds, then the following is performed for each match found.

4.2.2 R Instantiation

a copy of R is generated and its variables are instantiated according to their matching node in L. In addition, a rule may specify *alignments*, defined as a partial function from L nodes to R nodes. An alignment indicates that for each modifier m of the source node that is not part of the rule structure, the subtree rooted at m should also be copied as a modifier of the target node. In addition to explicitly defining alignments, each variable in L is implicitly aligned to its counterpart in R. In our example, the alignment between the V nodes implies that *yesterday* (modifying *see*) should be copied to the generated sentence, and similarly *little* (modifying *Mary*) is copied for $N1$.

4.2.3 Derived Tree Generation

Let r be the instantiated R, along with its descendants copied from L through alignment, and l be the subtree matched by L. The formalism has two methods for generating the derived tree d: *substitution* and *introduction*, as specified by the rule type. *Substitution* rules specify modification of a subtree of s, leaving the rest of s unchanged. Thus, d is formed by copying s while replacing l (and the descendants of l's nodes) with r. This is the case for the passive rule, as well as for lexical rules such as '*buy* $\rightarrow$ *purchase*'. By contrast, *introduction* rules are used to make inferences from a subtree of s, while the other parts of s are ignored and do not affect d. A typical example is inferring a proposition embedded as a relative clause in s. In this case, the derived tree d is simply taken to be

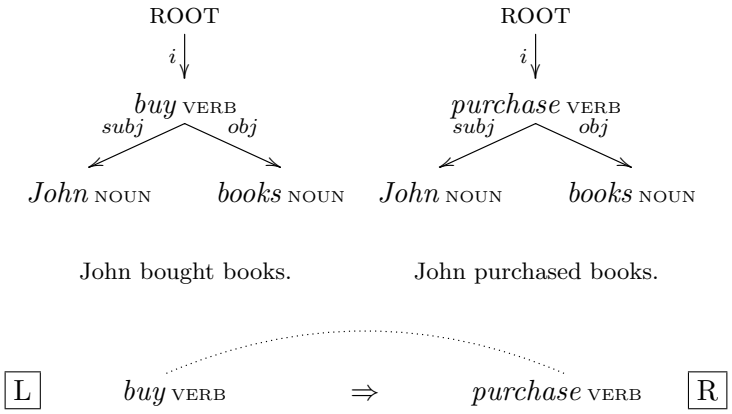


Figure 3: Application of a lexical substitution rule. The dotted arc represents explicit alignment.

r. Figure 2 presents such a rule, which enables deriving propositions that are embedded within temporal modifiers. Note that the derived tree does not depend on the main clause. Applying this rule to the right part of Figure 1a yields the proposition *"John saw little Mary yesterday"*.

4.3 Further Examples for Rule Application

In this section we further illustrate rule representation and application through additional examples.

4.3.1 Lexical Substitution Rule with Explicit Alignment

Figure 3 shows the derivation of the consequent *"John purchased books"* from the sentence *"John bought books"* using the lexical substitution rule '*buy* → *purchase*'. This example illustrates the role of explicit alignment: since *buy* and *purchase* are not variables, they are not implicitly aligned. However, they need to be aligned explicitly, otherwise the daughters of *buy* would not be copied under *purchase*.

4.3.2 Lexical-Syntactic Introduction Rule

Figure 4 illustrates the application of a lexical-syntactic rule, which derives the sentence *"Her husband died"* from *"I knew her late husband"*. It is defined as introduction rule, since the resulting tree is derived based solely on the phrase *"Her late husband"*, while ignoring the rest of the source tree. This example illustrates that a leaf variable in L (variable at a leaf node) may become a non-leaf in R and vice versa. The alignment between the instances of variable N (matched in *husband*) allows copying of its modifier, *her* (recall that such alignments are defined implicitly by the formalism). We note here that the correctness of rule application may depend on the context in which it is applied. For instance, the rule in our example is correct only if *late* has the meaning of "no longer alive" in the given context. We discuss context-sensitivity of rule application in Section 4.8.

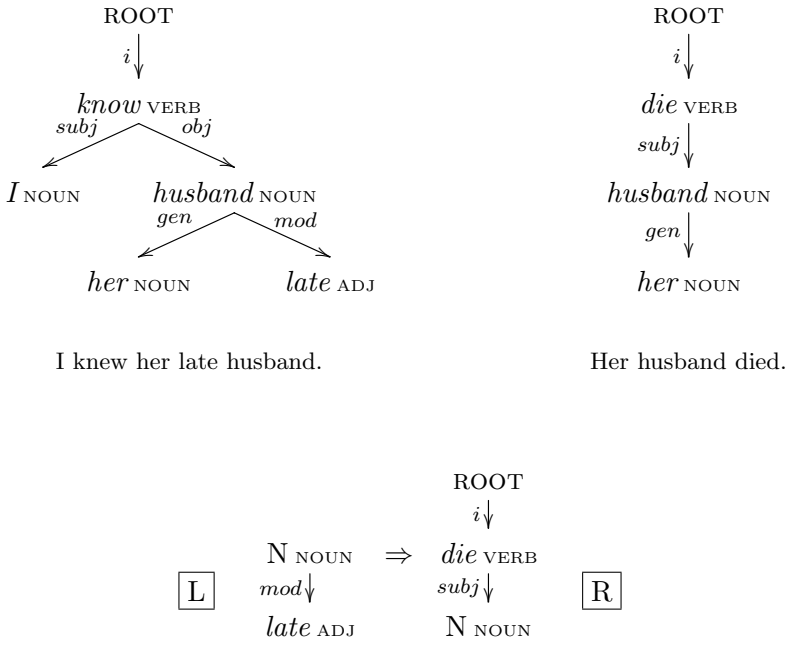


Figure 4: Application of a lexical-syntactic introduction rule

4.4 Co-Reference and Trace-Based Inference

Aside from the primary inference mechanism of rule application, our formalism also allows inference based on co-reference relations and long-distance dependencies. We view co-reference as an equivalence relation between complete subtrees, either within the same tree or in different trees, which are linked by a co-reference chain. In practice, such relations are obtained from an external co-reference resolution tool, as part of the text pre-processing. The *co-reference substitution* operation is similar to the application of a substitution rule. Given a pair of co-referring subtrees, t_1 and t_2, the derived tree is generated by copying the tree containing t_1, while replacing t_1 with t_2; the same operation is symmetrically applicable for t_2.[5] For example, given the sentences *"[My brother] is a musician. [He] plays the drums"*, we can infer that *"My brother plays the drums"*.

Long-distance dependencies are another type of useful relation for inference, as illustrated by the following examples:

(1) *Relative clause:* The boy$_i$ whom [I saw t$_i$] went home.
(⇒ I saw the boy.)

(2) *Control verbs:* John$_i$ managed to [t$_i$ open the door].
(⇒ John opened the door.)

5. The view of co-referring expressions as substitutional can also be found in the seminal paper of van Deemter and Kibble (2000), where noun phrases are shown to be non-substitutable as evidence that they are not co-referring.

(3) *Verbal conjunction:* [John$_i$ sang] and [t$_i$ danced].
(⇒ John danced.)

Some parsers including Minipar, which we use in the current work, recognize and annotate such long distance dependencies. For instance, Minipar generates a node representing the trace (t$_i$ in the examples), which holds a pointer to its antecedent (e.g., *John*$_i$ in (2)). As shown in these examples, inference from such sentences may involve resolving long- distance dependencies, where traces are substituted with their antecedent. Thus, we can generalize co-reference substitution to operate over trace-antecedent pairs, as well. This mechanism works together with inference rule application. For instance, after substituting the trace with its antecedent in (2) we obtain *"John managed to [John opened the door]"*. We then apply the introduction rule *'N managed to S → S'* to extract the embedded clause *"John opened the door"*.

4.5 Polarity Annotation Rules

In addition to inference rules, our formalism implementation includes a mechanism for adding semantic features to parse tree nodes. However, in many cases there is no natural way to define semantic features or classes. Hence, it is often difficult to agree on the "right" set of semantic annotations (a common example is the definition of word senses). With our approach, we aim to keep semantic annotation to a minimum, while sticking to lexical-syntactic representation, for which widely-agreed schemes do exist.

Consequently, the only semantic annotation we employ is predicate *polarity*. This feature marks the truth of a predicate, and may take one of the following values: *positive*(+), *negative*(-) or *unknown(?)*. Some examples of polarity annotation are shown below:

(4) John called$^{[+]}$ Mary.

(5) John *hasn't* called$^{[-]}$ Mary yet.

(6) John *forgot* to call$^{[-]}$ Mary.

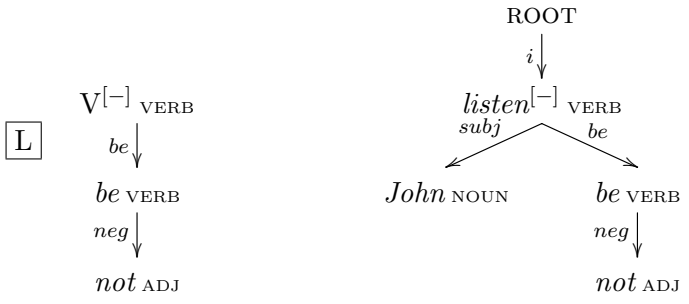
(7) John *might have* called$^{[?]}$ Mary.

(8) John *wanted* to call$^{[?]}$ Mary.

Sentences (5) and (6) both entail *"John didn't call Mary"*, hence the negative annotation of *call*. By contrast, the truth of *"John called Mary"* cannot be determined from (7) and (8), therefore the predicate *call* is marked as *unknown*. In general, the polarity of predicates may be affected by the existence of modals, negation, conditionals, certain verbs, etc.

Technically, annotation rules do not have a right-hand-side R, but rather each node of L may contain annotation features. If L is matched in a tree, then the annotations it contains are copied to the matched nodes. Figure 5 shows an example of annotation rule application.

Predicates are assumed to have positive polarity by default. The polarity rules are used to mark negative or unknown polarity. If more than one rule applies to the same predicate (as with the sentence *"John forgot not to call Mary"*), they may be applied in any order, and the following simple calculus is employed to combine current polarity with new polarity:



(a) Annotation rule (b) Annotated sentence

Figure 5: Application of the annotation rule (a), marking the predicate *listen* with negative polarity (b)

Current polarity	New polarity	Result
+	−	−
−	−	+
?	−	?
+/ − /?	?	?

Annotation rules are used for detecting polarity mismatches between the text and the hypothesis. Incompatible polarity would block the hypothesis from being matched in the text. In the case of approximate entailment classification, polarity mismatches detected by the annotation rules are used as features for the classifier, as we discuss further in Section 7.3. In addition, the existence of polarity annotation features may prevent inappropriate inference rule applications, by blocking their L matching. We discuss this further in Section 6.1.

4.6 The Inference Process

Let T be a set of dependency trees representing the text, along with co-reference and trace information. Let h be the dependency tree representing the hypothesis, and let $\mathcal{R}$ be a collection of inference rules (including both inference and polarity rules). Based on the previously defined components of our inference framework, we next give a procedural definition for the set of trees inferable from T using $\mathcal{R}$, denoted $\mathcal{I}(T, \mathcal{R})$. The inference process comprises the following steps:

1. Initialize $\mathcal{I}(T, \mathcal{R})$ with T.

2. Apply all matching polarity rules in $\mathcal{R}$ to each of the trees in $\mathcal{I}(T, \mathcal{R})$ (cf. Section 4.5).

3. Replace all the trace nodes with a copy of their antecedent subtree (cf. Section 4.4).

4. Add to $\mathcal{I}(T, \mathcal{R})$ all the trees derivable by co-reference substitution (cf. Section 4.4).

5. Apply all matching inference rules in $\mathcal{R}$ to the trees in $\mathcal{I}(T, \mathcal{R})$ (cf. Section 4.2), and add the derived trees to $\mathcal{I}(T, \mathcal{R})$. Repeat this step iteratively for the newly added trees, until no new trees are added.

Steps 2 and 3 are performed for h as well.[6] h is inferable from T using $\mathcal{R}$ if $h \in \mathcal{I}(T, \mathcal{R})$. Since $\mathcal{I}(T, \mathcal{R})$ may be infinite or very large, practical implementation of this process must limit the search space, for example by restricting the number of iterations and the applied rules at each iteration.

When an inference rule is applied, polarity annotation is propagated from the source tree s to the derived tree d as follows. First, nodes copied from s to d retain their original polarity. Second, a node in d gets the polarity of its aligned node in s.

4.7 Template Hypotheses

For many applications it is useful to allow the hypothesis h to be a template rather than a proposition, that is, to contain variables. The variables in this case are existentially quantified: t entails h if there exists a proposition h', obtained from h by variable instantiation, so that t entails h'. Each variable X is instantiated (replaced) with a subtree S_X. If X has modifiers in h (i.e., X is not a leaf), they become modifiers of S_X's root. The obtained variable instantiations may stand for answers sought in questions or slots to be filled in relation extraction. For example, applying this framework in a question-answering setting, the question *Who killed Kennedy?* may be transformed into the hypothesis *X killed Kennedy*. A successful proof of h from the sentence *"The assassination of Kennedy by Oswald shook the nation"* would instantiate X with *Oswald*, providing the sought answer.

4.8 Limitations and Possible Extensions

We conclude this section by discussing some limitations of the presented inference formalism, as well as possible extensions to address these limitations. First, our inference rules match only a single subtree, and therefore are less expressive than the logic axioms used by Bos and Markert (2005) and Tatu and Moldovan (2006), which may combine several predicates originating from the text representation as well as from the background knowledge. This allows logic-based systems to make inferences that combine multiple pieces of information. For instance, if the text says that a person X lives in a city Y, and the background knowledge tells us that the city Y is in country Z, we can infer that X lives in country Z, using a rule such as *"person(X)* $\wedge$ *location(Y)* $\wedge$ *location(Z)* $\wedge$ *live(X,Y)* $\wedge$ *in(Y,Z)* $\rightarrow$ *live(X,Z)"* . Schoenmackers, Etzioni, Weld, and Davis (2010) describe a system that acquires such rules (first-order horn clauses) from Web text. Allowing our rules to match multiple subtrees in t, as well as information in the background knowledge, seems a plausible future extension to our formalism.

Another limitation of the formalism is the lack of context disambiguation. Word sense mismatch is a potential cause for incorrect rule applications. For example, the rule *'hit* $\rightarrow$ *score'* is applied correctly in (9) but not in (10):

6. Step 4 is not applied to h since the hypothesis is typically a short, simple sentence that usually does not include co-referring NPs. Moreover, in the presented formalism h is a single tree. Applying co-reference-based inference would have resulted in additional trees inferred from h, and thus would have required extending the formalism accordingly.

(9) The team hit a home run. ⇒ The team scored a home run.

(10) The car hit a tree. ⇏ The car scored a tree.

Several works over the past years addressed the problem of context-dependent rule application (Dagan, Glickman, Gliozzo, Marmorshtein, & Strapparava, 2006a; Pantel, Bhagat, Coppola, Chklovski, & Hovy, 2007; Connor & Roth, 2007; Szpektor, Dagan, Bar-Haim, & Goldberger, 2008; Dinu & Lapata, 2010; Ritter, Mausam, & Etzioni, 2010; Berant, Dagan, & Goldberger, 2011; Melamud, Berant, Dagan, Goldberger, & Szpektor, 2013). Szpektor et al. (2008) proposed a comprehensive framework for modeling context matching, termed *Contextual Preferences (CP)*. Given a text t, a hypothesis h (possibly a template hypothesis) and an inference rule r bridging between t and h, each of these objects is annotated with two context components: (a) global ("topical") context, and (b) preferences and constraints on the instantiation of the object's variables (for r and template h). CP requires that h and r are matched in t, and h is matched in r[7], where each context component is matched to its counterpart. Szpektor et al. also proposed concrete implementations for each of these components. In the above example, we could model the global context of t and r as the sets of their content words, and compute the semantic relatedness between these two sets, using methods such as Latent Semantic Analysis (LSA) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990), or Explicit Semantic Analysis (ESA) (Gabrilovich & Markovitch, 2007). We would expect that the semantic relatedness between {*score*} and {*team, home run*} will be much higher than between {*score*} and {*car, tree*}, which would permit inference in (9) but not in (10).

In most RTE systems (including our system in the RTE experiments, described in Section 7.3) lexicalized rules bridge between t and h directly, so that the rule's LHS and RHS are matched in t and h, respectively. Since in the RTE benchmarks t and h tend to have the same semantic context, this setting alleviates context matching problems to some extent. However, our analysis, presented later in this work (Subsection 7.5.2), shows that context matching remains an issue even in this setting, and is expected to become even more important when chaining of lexicalized rules is attempted. Adding contextual preferences to our formalism is an important direction for future work.

The validity of rule application also depends on the monotonicity properties of its application site. For instance, the hypernym rule *poodle* → *dog* is applicable only in *upward monotone* contexts. Monotonicity may be affected by the presence of quantifiers, negation, and certain verbs such as implicatives and counterfactives (Nairn, Condoravdi, & Karttunen, 2006). As common with textual entailment systems, we assume upward monotonicity anywhere. While this assumption usually holds true, in some cases it may lead to incorrect inferences. The following examples show correct applications of the above rule in upward monotone contexts ((11),(14)), and incorrect applications in downward monotone contexts ((12),(13),(15)):

(11) She bought a poodle. ⇒ She bought a dog.

(12) She didn't buy a poodle ⇏ she didn't buy a dog

(13) Poodles are smart. ⇏ Dogs are smart.

7. Context matching, like textual entailment, is a directional relation.

(14) She failed to avoid buying a poodle ⇒ She failed to avoid buying a dog.

(15) She did not fail to avoid buying a poodle ⇏ She did not fail to avoid buying a dog.

MacCartney and Manning (2009) address monotonicity as well as other semantic relations such as exclusion, in a Natural Logic framework based on syntactic representation. We discuss their work in more detail in Section 8.

Finally, since our polarity annotation rules are applied locally, they may fail in complex cases, such as computing the polarity of *buying* in sentences (14) and (15), in which polarity information need to be propagated along the syntactic structure of the sentence. The *TruthTeller* system (Lotan, Stern, & Dagan, 2013), computes predicate polarity (truth value) by a combination of annotation rules and a global polarity propagation algorithm, extending previous work by Nairn et al. (2006) and MacCartney and Manning (2009).

4.9 Summary

In this section, we presented a well-formalized approach for textual inference over parse-based representations, which is the core of this paper. In our framework, semantic knowledge is represented uniformly as inference rules specifying tree transformations. We provided detailed definitions for the representation of these rules as well as the inference mechanisms that apply them. Our formalism also models inferences based on co-reference relations and traces. In addition, it includes *annotation* rules that are used to detect contexts affecting the polarity of predicates. In the next section we present an efficient implementation of this formalism.

5. A Compact Forest for Scalable Inference

According to our formalism, each rule application generates a new sentence parse (a *consequent*), semantically entailed by the source sentence. Each inferred consequent may be subject to further rule applications, and so on. A straightforward implementation of this formalism would generate each consequent as a separate tree. Unfortunately, this naïve approach raises severe efficiency issues, since the number of consequents may grow exponentially in the number of rule applications. Consider, for example, the sentence *"Children are fond of candies"*, and the following rules: *'children→kids'*, *'candies→sweets'*, and *'X is fond of Y→X likes Y'*. The number of derivable sentences, including the source sentence, would be 2^3 (the power set size), as each rule can either be applied or not, independently. We found that this exponential explosion leads to poor scalability of the naïve implementation approach in practice.

Intuitively, we would like for each rule application to add just the entailed part of the rule (e.g., *kids*) to a packed sentence representation. Yet, we still want the resulting structure to represent a set of entailed sentences, rather than a mixture of sentence fragments with unclear semantics. As discussed in Section 8, previous work proposed only partial solutions to this problem.

In this section, we introduce a novel data structure, termed *compact forest*, and a corresponding inference algorithm, which efficiently generate and represent all consequents while preserving the identity of each individual one. This data structure allows compact representation of a large set of inferred trees. Each rule application generates explicitly only the

nodes of the rule's right-hand-side. The rest of the consequent tree is shared with the source sentence, which also reduces the number of redundant rule applications, as explained later in this section. We show that this representation is based primarily on *disjunction edges*, an extension of dependency edges that specify a set of alternative edges of multiple trees.

Since we follow a well-defined inference formalism, we are able to prove that all inference operations in our formalism are equivalently applied over the compact forest. We compare inference cost over compact forests to explicit consequent generation both theoretically, illustrating an exponential-to-linear complexity ratio, and empirically, showing improvement by orders of magnitude (empirical results are reported in Section 7.2).

5.1 The Compact Forest Data Structure

A compact forest $\mathcal{F}$ represents a set of dependency trees. Figure 6d shows an example of a compact forest containing trees for the sentences *"Little Mary was seen by John yesterday"* and *"John saw little Mary yesterday"*. We first define a more general data structure for directed graphs, and then narrow the definition to the case of trees.

A *Compact Directed Graph (cDG)* is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of **disjunction edges** *(d-edges)*. Let $\mathcal{D}$ be a set of dependency relations. A d-edge d is a triple (S_d, rel_d, T_d), where S_d and T_d are disjoint sets of source nodes and target nodes; $rel_d : S_d \to \mathcal{D}$ is a function specifying the dependency relation that corresponds to each source node. Graphically, d-edges are shown as point nodes, with incoming edges from source nodes and outgoing edges to target nodes. For instance, let d be the bottommost d-edge in Figure 7. Then $S_d = \{of, like\}$, $T_d = \{candy, sweet\}$, $rel(of) = pcomp\text{-}n$, and $rel(like) = obj$.

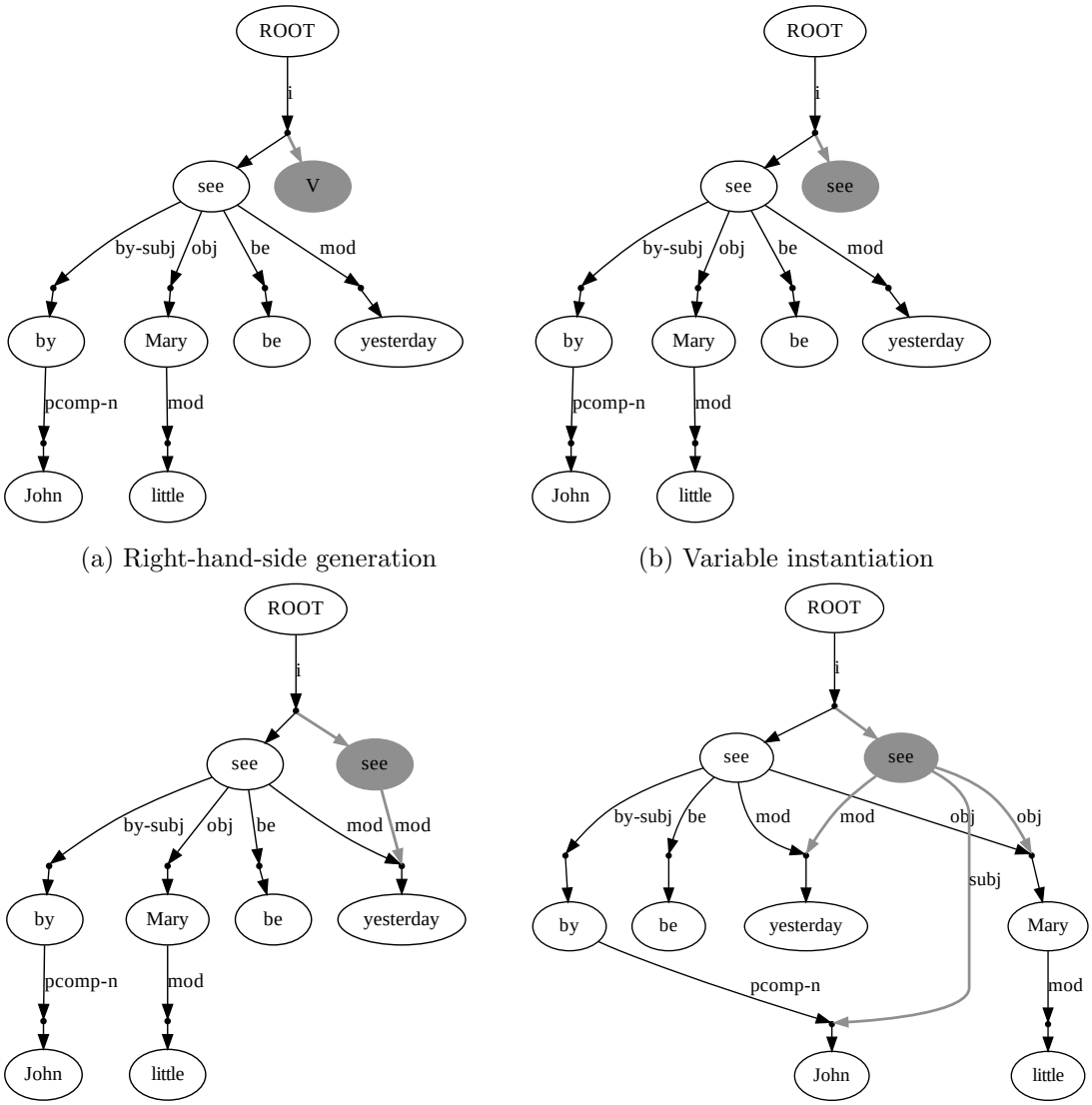
A d-edge represents, for each $s_i \in S_d$, a set of alternative directed edges $\{(s_i, t_j) : t_j \in T_d\}$, all of which are labeled with the same relation given by $rel_d(s_i)$. Each of these edges, termed *embedded edge (e-edge)*, would correspond to a different graph represented in $\mathcal{G}$. In the previous example, the e-edges are $like \xrightarrow{obj} candy$, $like \xrightarrow{obj} sweet$, $of \xrightarrow{pcomp-n} candy$ and $of \xrightarrow{pcomp-n} sweet$ (the definition implies that all source nodes in S_d have the same set of alternative target nodes T_d). The d-edge d is called an *outgoing d-edge* of a node v if $v \in S_d$ and an *incoming d-edge* of v if $v \in T_d$. A *Compact Directed Acyclic Graph (cDAG)* is a cDG that contains no cycles of e-edges.

A DAG G rooted in a node $v \in \mathcal{V}$ of a cDAG $\mathcal{G}$ is *embedded* in $\mathcal{G}$ if it can be derived as follows: we initialize G with v alone; then, we *expand* v by choosing exactly one target node $t \in T_d$ from each outgoing d-edge d of v, and adding t and the corresponding e-edge (v, t) to G. This expansion process is repeated recursively for each new node added to G.

Each such set of choices results in a different DAG with v as its only root. In Figure 6d, we may choose to connect the root either to the left *see*, resulting in the source passive sentence, or to the right *see*, resulting in the derived active sentence.

A **Compact Forest** $\mathcal{F}$ is a cDAG with a single root r (i.e., r has no incoming d-edges) where all the embedded DAGs rooted in r are trees. This set of trees, termed *embedded trees*, and denoted $\mathcal{T}(\mathcal{F})$ comprise the set of trees represented by $\mathcal{F}$.

Figure 7 shows another example of a compact forest efficiently representing the 2^3 sentences resulting from the three independently applied rules presented at the beginning of this section.



(a) Right-hand-side generation

(b) Variable instantiation

(c) Alignment sharing

(d) Dual-leaf variable sharing

Figure 6: Step-by-step construction of the compact forest containing both the source sentence *"Little Mary was seen by John yesterday"* and the sentence *"John saw little Mary yesterday"* derived from it via the application of the passive rule of Figure 1b. Parts of speech are omitted.

5.2 The Inference Process

We next describe the algorithm implementing the inference process described in Section 4.6 over the compact forest (henceforth, *compact inference*), illustrated by Figures 1b (the passive-to-active rule) and 6.

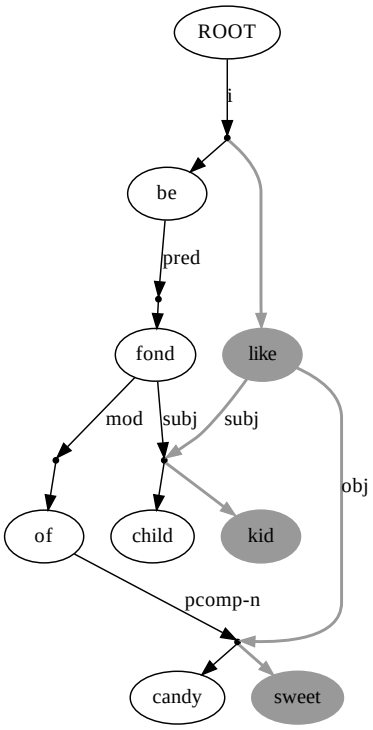


Figure 7: A compact forest representing the 2^3 sentences derivable from the sentence *"Children are fond of candies"* using the following three rules: *'children→kids'*, *'candies→sweets'*, and *'X is fond of Y→X likes Y'*.

5.2.1 Forest Initialization

$\mathcal{F}$ is initialized with the set of dependency trees representing the *text* sentences, with their roots connected under the forest root as the target nodes of a single d-edge. Dependency edges are transformed trivially to d-edges with a single source and target. Annotation rules are applied at this stage to the initial $\mathcal{F}$. Figure 6a, without the node labeled "V" and its incoming edge, corresponds to the initial forest (containing a single sentence in our example).

5.2.2 Inference Rule Application

Inference rule application comprises the steps described below, which are summarized in Algorithm 2.

L Matching We first find all the matches of the rule's LHS L in the forest $\mathcal{F}$ (line 1). For the sake of brevity, we omitted the technical details of the L matching implementation from the pseudocode of Algorithm 2. The following is a high-level description of the matching procedure, focusing on the key algorithmic points.

L is matched in $\mathcal{F}$ if there exists an embedded tree t in $\mathcal{F}$ such that L is matched in t, as in Section 4.2. We denote by l the subtree of t in which L was matched (line 3).

Input: a compact forest $\mathcal{F}$; an inference rule $E : L \rightarrow R$
Output: A modified $\mathcal{F}$, denoted $\mathcal{F}'$, such that $\mathcal{T}(\mathcal{F}') = \mathcal{T}(\mathcal{F}) \cup D$, where D is the set of trees derived by applying E for any subset of L's matches in each of the trees in $\mathcal{T}(\mathcal{F})$

$M \leftarrow$ the set of all matches of L in $\mathcal{F}$

for each match $f \in M$ **do**
  $l \leftarrow$ the subtree of $\mathcal{F}$ in which L is matched according to f

  // Right-hand-side generation
  $S_R \leftarrow$ copy of R excluding *dual leaf variable* nodes
  Add S_R to $\mathcal{F}$
  $S_L \leftarrow l$ excluding *dual leaf variable* nodes
  $r_R \leftarrow root(S_R)$
  $r_L \leftarrow root(l)$
  if E is a *substitution* rule **then**
    $d \leftarrow$ the incoming d-edge of r_L // will set S_R as an alternative to S_L
  else // *introduction* rule
    $d \leftarrow$ the outgoing d-edge of $root(\mathcal{F})$ // will set S_R as an alternative to other trees in $\mathcal{T}(\mathcal{F})$
  Add r_R to T_d

  // Variable instantiation
  for each variable X held in node $x_R \in S_R$ **do** // R's variables excluding dual leaves
    if X is not a leaf in L **then**
      $x_L \leftarrow f(X)$ // the node in S_L matched by X
      (x_R.lemma, x_R.polarity) $\leftarrow$ (x_L.lemma, x_L.polarity)

    else // X is a leaf in L so it is matched in the whole target node set
      (x_R.lemma, x_R.polarity) $\leftarrow$ (n.lemma, n.polarity) for some node $n \in f(X)$
      for each $n' \in f(X); n' \neq n$ **do**
        generate a substitution rule $n \rightarrow n'$ where n and n' are aligned, and apply it to x_R
        $x'_R \leftarrow$ the instantiation of n'
        for each $u \in S_L$ such that u is aligned to x_R **do**
          add alignment from u to x'_R

  // Alignment sharing
  for each aligned pair of nodes $n_L \in S_L$ and $n_R \in S_R$ **do**
    n_R.polarity $\leftarrow$ n_L.polarity
    for each outgoing d-edge d of n_L whose e-edges are not part of S_L **do**
      Add n_R to S_d
      $rel_d(n_R) \leftarrow rel_d(n_L)$

  // Dual leaf variable sharing
  for each dual-leaf variable X matched in a node $v \in l$ **do**
    $d \leftarrow$ the incoming d-edge of v
    $p \leftarrow$ parent node of X in S_R

    // go over p and alternatives for p generated during variable instantiation
    $P \leftarrow$ set of target nodes of p's incoming d-edge
    for each $p' \in P$ **do**
      Add p' to S_d
      $rel_d(p') \leftarrow$ the relation between X and p

Algorithm 2: Applying an inference rule to a compact forest

This subtree may be shared by multiple trees represented in $\mathcal{F}$, in which case the rule is applied simultaneously to all of these trees. As in Section 4.2, the match in our example is $(V, N1, N2)$=*(see, Mary, John)*. This definition does not allow l to be scattered over multiple embedded trees. Matches are constructed incrementally, aiming to add L's nodes one by one to the partial matches constructed so far, while verifying for each candidate node in $\mathcal{F}$ that both node content and the corresponding edge labels match. It is also verified that the match does not contain more than one e-edge from each d-edge. The nodes in $\mathcal{F}$ are indexed using a hash table to enable fast lookup.

As the target nodes of a d-edge specify alternatives for the same position in the tree, their parts-of-speech are expected to be substitutable. We further assume that all target nodes of the same d-edge have the same part-of-speech[8] and polarity. Consequently, variables that are leaves in L and may match a certain target node of a d-edge d are mapped to the whole set of target nodes T_d rather than to a single node. This yields a compact representation of multiple matches, and prevents redundant rule applications. For instance, given a compact representation of *'{Children/kids} are fond of {candies/sweets}'* (cf. Figure 7), the rule *'X is fond of Y→X likes Y'* will be matched and applied only once, rather than four times (for each combination of matching X and Y).

Right-Hand-Side Generation Given an inference rule $L \rightarrow R$, we define a *dual-leaf variable* as a variable that is a leaf of both L and R. In our example, both $N1$ and $N2$ are dual-leaf variables in the passive-to-active rule of Figure 1b. Variables that are the only node in R (and hence are both the root and a leaf), and variables with additional alignments (other than the implicit alignment between their occurrences in L and R) are not considered dual-leaves. As explained below, the instantiations of dual leaf variables are shared between the source and the target trees.

In the *right-hand-side generation* step, a template S_R (line 5), consisting of R while excluding dual-leaf variables, is generated and inserted into $\mathcal{F}$ (line 6). In our example, S_R only includes the node V out of the passive rule's RHS. Similarly, we define S_L as l excluding dual-leaf variables (line 7).

In the case of a substitution rule (as in our example), S_R is set as an alternative to S_L by adding S_R's root to T_d, where d is the incoming d-edge of S_L's root (line 11). In case of an introduction rule, it is set as an alternative to the other trees in the forest by adding S_R's root to the target node set of the forest root's outgoing d-edge (line 13). Figure 6a illustrates the results of this step for our example. S_R is the gray node labeled with the variable V, and it becomes an additional target node of the d-edge entering the original (left) *see*.

Variable Instantiation Each variable in S_R (i.e., a non dual-leaf) is instantiated (lines 16-26) according to its match in L (as in Section 4.2). In our example, V is instantiated with *see* (Figure 6b, lines 17-19). As specified above, if a variable in S_R is a leaf in L (which is not the case in our example) then it is matched in a set of nodes, and each of them should be instantiated in S_R (lines 20-26). This is decomposed into a sequence of simpler operations: first, S_R is instantiated with a representative from the set (line 21). We then apply ad-hoc lexical substitution rules for creating a new node for each additional node in

8. This is the case in our current implementation, which is based on the coarse tag-set of Minipar.

the set (line 22-26). These nodes, in addition to the usual alignment with their source nodes in S_L (lines 25-26), share the same daughters in S_R (due to the alignment between n and n', defined in line 23).

Alignment Sharing Modifiers of aligned nodes are shared (rather than copied) as follows. Given a node n_L in S_L aligned to a node n_R in S_R, and an outgoing d-edge d of n_L which is not part of l, we share d between n_L and n_R by adding n_R to S_d and setting $rel_d(n_R) = rel_d(n_L)$ (lines 28-32). In our example (Figure 6c), the aligned nodes n_L and n_R are the left and right *see* nodes, respectively, and the shared modifier is *yesterday*. The dependency relation *mod* is copied for the right *see* node. We also copy polarity annotation from n_L to n_R (line 29).

We note at this point that the instantiation of variables that are not dual leaves cannot be shared because they typically have different modifiers at the two sides of the rule. Yet, their modifiers, which are not part of the rule, are shared through the alignment operation (recall that common variables are always considered aligned). Dual leaf variables, on the other hand, might be shared, as described next, since the rule doesn't specify any modifiers for them.

Dual Leaf Variable Sharing This final step (lines 34-41) is performed similarly to alignment sharing. Suppose that a dual leaf variable X is matched in a node v in l whose incoming d-edge is d. Then we simply add the parent p of X in S_R to S_d and set $rel_d(p)$ to the relation between p and X (in R). Since v itself is shared, its modifiers become shared as well, implicitly implementing the alignment operation. The subtrees *little Mary* and *John* are shared this way for variables $N1$ and $N2$ (Figure 6d). If ad-hoc substitution rules were applied to p at the variable instantiation phase, the generated nodes serve as alternative parents of X, thus the sharing procedure applied to p should be repeated for each of them.

Applying the rule in our example added only a single node and linked it to four d-edges, compared to duplicating the whole tree in explicit inference.

5.2.3 Co-reference Substitution

In Section 4.4 we defined *co-reference substitution*, an inference operation that allows replacing a subtree t_1 with a co-referring subtree t_2. This operation is implemented by generating on-the-fly a substitution rule $t_1 \rightarrow t_2$ and applying it to t_1. In our implementation, the initial compact forest is annotated with co-reference relations obtained from an external co-reference resolution tool, and all substitutions are performed prior to rule applications. Substitutions where t_2 is a pronoun are ignored, as they are usually not useful.

5.3 Correctness

In this section, we present two theorems proving that the inference process presented is a valid implementation of the inference formalism. We provide the full proofs in Appendix A.

In Theorem 1, we argue that applying a rule to a compact forest results in a compact forest. Since we begin with a valid compact forest created by the initialization step, it follows by induction that for any sequence of rule applications the result of the inference process is a compact forest. The fact that the embedded DAGs generated during the inference process are indeed trees is not trivial, since nodes generally have many incoming e-edges

from many nodes. However, we show that any pair of these parent nodes cannot be part of the same embedded DAG. For example, in Figure 7, the node *'candy'* has an incoming e-edge from both the node *'like'* and the node *'of'*. However, the nodes *'like'* and *'of'* are not part of the same embedded DAG. This is because the d-edge emanating from the root forces us to choose between the node *'like'* and the node *'be'*. Thus, we see that the reason for correctness is not local: the two incoming e-edges into the leaf node *'candies'* cannot be in the same embedded DAG because of a rule applied at the root of the tree. We now turn to the theorem and its proof scheme:

Theorem 1 *Applying a rule to a compact forest results in a compact forest.*

Proof scheme We prove that if applying a rule to a compact forest creates a cycle or an embedded DAG that is not a tree, then such a cycle or a non-tree DAG already existed prior to rule application. This contradicts the assumption that the original structure is a compact forest. A crucial observation for this proof is that for any directed path from a node u to a node v that passes through S_R, where u and v are outside S_R, there is also an analogous path from u to v that passes through S_L instead. ∎

The next theorem is the main result. We argue that the inference process over a compact forest is complete and sound, that is, it generates exactly the set of consequents derivable from a text according to the inference formalism.

Theorem 2 *Given a rule base $\mathcal{R}$ and a set of initial trees T, a tree t is represented by a compact forest derivable from T by the inference process $\Leftrightarrow$ t is a consequent of T according to the inference formalism.*

Proof scheme We first show completeness by induction on the number of explicit rule applications. Let t_{n+1} be a tree derived from a tree t_n using the rule r_n according to the inference formalism. The inductive assumption determines that t_n is embedded in some derivable compact forest $\mathcal{F}$. It is easy to verify that applying r_n on $\mathcal{F}$ will yield a compact forest $\mathcal{F}'$ in which t_{n+1} is embedded.

Next, we show soundness by induction on the number of rule applications over the compact forest. Let t_{n+1} be a tree represented in some derived compact forest $\mathcal{F}_{n+1}$ ($t_{n+1} \in \mathcal{T}(\mathcal{F}_{n+1})$). $\mathcal{F}_{n+1}$ was derived from the compact forest $\mathcal{F}_n$, using the rule r_n. The inductive assertion states that all the trees in $\mathcal{T}(\mathcal{F}_n)$ are consequents of T according to the formalism. Hence, if t_{n+1} is already in $\mathcal{T}(\mathcal{F}_n)$ then it is a consequent of T. Otherwise, it can be shown that there exists a tree $t_n \in \mathcal{T}(\mathcal{F}_n)$ such that applying r_n to t_n will yield t_{n+1} according to the formalism. t_n is a consequent of T according to the inductive assertion and therefore t_{n+1} is a consequent of T as well. ∎

These two theorems guarantee that the compact inference process is valid, that is, it yields a compact forest that represents exactly the set of consequents derivable from a given text by a given rule set.

5.4 Complexity

In this section, we explain why compact inference exponentially reduces the time and space complexity in typical scenarios.

We consider a set of rule matches in a tree T *independent* if their matched left-hand-sides (excluding dual-leaf variables) do not overlap in T, and their application over T can be chained in any order. For example, the three rule matches presented in Figure 7 are independent.

Let us consider explicit inference first. Assume we start with a single tree T with k independent rules matched. Applying k rules will yield 2^k trees, since any subset of the rules might be applied to T. Therefore, the time and space complexity of applying k independent rule matches is $\Omega(2^k)$. Applying more rules on the newly derived consequents behaves in a similar manner.

Next, we examine compact inference. Applying a rule using compact inference adds the right-hand-side of the rule and shares with it existing d-edges. Since the size of the right-hand-side and the number of outgoing d-edges per node are practically bounded by low constants, applying k rules on a tree T yields a linear increase in the size of the forest. Thus, the resulting size is $O(|T| + k)$, as we can see from Figure 7.

The time complexity of rule application is composed of matching the rule in the forest and applying the matched rule. Applying a matched rule is linear in its size. Matching a rule of size r in a forest $\mathcal{F}$ takes $O(|\mathcal{F}|^r)$ time even when performing an exhaustive search for matches in the forest. Since r tends to be quite small and can be bounded by a low constant[9], this already gives polynomial time complexity. Furthermore, matches are constructed incrementally, where at each step we aim to extend the partial matches found. Due to the typical low connectivity of the forest, as well as the various constraints imposed by the rule (lemma, POS, and dependency relation), the number of candidates for extending the matches at each step $<< |\mathcal{F}|$, and these candidates can be retrieved efficiently using proper indexing. Thus, the matching procedure is very fast in practice, as illustrated in the empirical evaluation described in Section 7.2.

5.5 Related Work on Packed Representations

Packed representations in various NLP tasks share common principles, which also underlie our compact forest: factoring out common substructures and representing choice as local disjunctions. Applying this general scheme to individual problems typically requires specific representations and algorithms, depending on the type of alternatives that should be represented and the specified operations for creating them. We create alternatives by rule application, where a newly derived subtree is set as an alternative to existing subtrees. Alternatives are specified locally using d-edges.

Packed chart representations for parse forests were introduced in classical parsing algorithms such as CYK and Earley (Jurafsky & Martin, 2008), and were extended in later work for various purposes (Maxwell III & Kaplan, 1991; Kay, 1996). Alternatives in the parse chart stem from syntactic ambiguities, and are specified locally as the possible decompositions of each phrase into its sub-phrases.

9. In our RTE system, the average rule LHS size was found to be 2 nodes, and the maximal size was 7 nodes, for the experimental setting described in Section 7.2.2, applied to the RTE3 test set.

Packed representations have also been utilized in transfer-based machine translation. Emele and Dorna (1998) translated packed source language representation to packed target language representation while avoiding unnecessary unpacking during transfer. Unlike our rule application, in their work transfer rules *preserve* ambiguity stemming from source language, rather than *generating* new alternatives. Mi et al. (2008) applied statistical machine translation to a source language parse forest, rather than to the 1-best parse. Their transfer rules are tree-to-string, contrary to our tree-to-tree rules, and chaining is not attempted (rules are applied in a single top-down pass over the source forest). Thus, their representation and algorithms are quite different from ours.

6. Incorporated Knowledge Bases

In this section, we describe the various knowledge bases used by our inference engine. We first describe a novel rule base addressing generic linguistic structures. This rule base was composed manually, based on our formalism, and includes both inference rules (Section 6.1) and polarity annotation rules (Section 6.2). In addition, we derived inference rules from several large scale semantic resources (Section 6.3). Overall, this variety illustrates the suitability of our formalism for representing diverse types of inference knowledge.

6.1 Inference Rules for Generic Linguistic Phenomena

These rules capture inferences associated with common syntactic structures, which are summarized in Table 2. The rules have three major functions:

1. Simplification and canonization of the source tree (categories 6 and 7 in Table 2).
2. Extracting embedded propositions (categories 1, 2, 3).
3. Inferring propositions from non-propositional subtrees of the source tree (category 4).

Inference rules that merely extract a subtree out of the source tree without changing its structure (such as the relative clause rule) are useful for exact inference that aims to generate the hypothesis, and were used in the evaluation of such inferences (cf. Section 7.1). However, the currently implemented approximate classification features are focused on matching substructures of the hypothesis in the forest (as described in Section 7.3), hence they do not take advantage of such extractions. Therefore, these rules were excluded from the rest of the experiments, reported in Sections 7.2–7.3.

The rules in categories 1-7 depend solely on syntactic structure and closed-class words, and are referred to as *generic rules*. By contrast, verb complement extraction rules (category 8) are considered *lexicalized rules*, since they are specific to certain verbs: if we replace *forced* with *advised* in the example, the entailment would not hold. We extracted from the PARC polarity lexicon (Nairn et al., 2006) a list of verbs that allow such inference when appearing in positive polarity contexts, and generated inference rules for these verbs. The list was complemented with a few reporting verbs, such as *say* and *announce*, since information in the news domain, in which these rules were applied in our experiments (cf. Section 7.1) is often given in reported speech, while the speaker is usually considered reliable.

We sidestep the issue of polarity propagation by applying these rules only at the main clause, which is implemented by including the tree root node in the rule LHS. When the

#	Category	Example: source	Example: derived
1	Conjunctions	Helena's very experienced **and** has played a long time on the tour.	⇒ Helena has played a long time on the tour.
2	Clausal extraction from connectives	But celebrations were muted **as** many Iranians observed a Shi'ite mourning month.	⇒ Many Iranians observed a Shi'ite mourning month.
3	Relative clauses	The assailants fired six bullets at the car, **which** carried Vladimir Skobtsov.	⇒ The car carried Vladimir Skobtsov.
4	Appositives	**Frank Robinson, a one-time manager of the Indians**, has the distinction for the NL.	⇒ Frank Robinson is a one-time manager of the Indians.
5	Determiner Canonization	The plaintiffs filed **their** lawsuit last year in U.S. District Court in Miami.	⇒ The plaintiffs filed **a** lawsuit last year in U.S. District Court in Miami.
6	Passive	We **have been approached by** the investment banker.	⇒ The investment banker **approached** us.
7	Genitive modifier	**Malaysia's crude palm oil output** is estimated to have risen by up to six percent.	⇒ **The crude palm oil output of Malaysia** is estimated to have risen by up to six percent.
8	Verb complement clause extraction	Yadav **was forced to** resign.	⇒ Yadav resigned.

Table 2: Inference rules for generic linguistic structures

embedded clause is extracted, it becomes the main clause in the derived tree, and these rules can then extract its own embedded clauses. The polarity of the verb is detected by applying annotation rules, as described next. If the verb was annotated with negative or unknown polarity, matching of complement extraction rules fails. For example, if the last sentence in Table 2 was "Yadav was *not* forced to resign", then *forced* would be annotated with negative polarity, and consequently the matching of the corresponding complement extraction rule would fail, and "Yadav resigned" would not be entailed. Hence, annotation rules may block erroneous inference rule applications. While polarity is important for correct application of such rules, this is not the case for other rule types, such as passive-to-active transformation. We therefore checked polarity matching for rule application only in the exact inference experiment (Section 7.1), where the verb complement extraction rules were used. We leave further analysis of polarity-dependence of our rules to future work.

6.2 Polarity Annotation Rules

We use annotation rules to mark negative and unknown polarity of predicates (cf. Section 4.5). Table 3 summarizes the polarity-inducing contexts that we address. Like inference rules, annotation rules also comprise *generic* rules (categories 1-4) and *lexicalized*

#	Category	Example
1	Explicit Negation	What we've **never** seen$^{[-]}$ is actual costs come down.
2	Implied Negation	**No one** stayed$^{[-]}$ for the last lecture.
3	Modal Auxiliaries	I **could** eat$^{[?]}$ a whale now!
4	Overt Conditionals	**if** Venus wins$^{[?]}$ this game, she will meet$^{[?]}$ Sarena in the finals.
5	Verb complements	I **pretend** that I know$^{[-]}$ calculus.
6	Adjectives	It is **impossible** that he survived$^{[-]}$ such a fall.
7	Adverbs	She **probably** danced$^{[?]}$ all night.

Table 3: Polarity annotation rules

rules (categories 5-7). When a verb complement embedded clause has negative or unknown polarity, it is not extracted, however, its polarity is annotated (category 5; compare with category 8 in Table 2). This list of verbs that imply negative/unknown polarity for their clausal complements was taken from the PARC lexicon, as well as from VerbNet (Kipper, 2005).

6.3 Lexical and Lexical-Syntactic Rules

In addition to the manually-composed generic rules, the system integrates inference knowledge from a variety of large-scale semantic resources, introduced in Section 2.3. The information derived from these resources is represented uniformly as inference rules in our formalism. Some examples for such rules were shown in Table 1. The following resources were used:

WordNet: We extracted from WordNet (Fellbaum, 1998) lexical rules based on the *synonym*, *hyponym* (a word is entailed by its hyponym, e.g., '*dog*→ *animal*'), *instance hyponym*[10] and *derivation* relations.

Wikipedia: We used the lexical rulebase of Shnarch et al. (2009), who extracted rules such as *'Janis Joplin → singer'* from Wikipedia based on both its metadata (e.g., links and redirects) and text definitions, using patterns such as '*X is a Y*'.[11]

DIRT: The DIRT algorithm (Lin & Pantel, 2001) learns from a corpus inference rules between binary predicates, for example, *'X is fond of Y*→*X likes Y'*. We used a version that learns canonical rule forms (Szpektor & Dagan, 2007).

Argument-Mapped WordNet (AmWN): A resource for inference rules between predicates, covering both verbal and nominal forms (Szpektor & Dagan, 2009), includ-

10. According to the WordNet glossary, an *instance* is a proper noun that refers to a particular, unique referent (as distinguished from nouns that refer to classes). This is a specific form of hyponym. For example, *Ganges* is an instance of *river*.

11. In addition to the extraction methods described by Shnarch et al. (2009), we employed two additional methods. First, extraction of entailments among terms that are redirected to the same page. Second, generalization of rules with the same RHS and common LHS head, but different modifiers. For instance, the rules '*Ferrari F430*→ *car*' and '*Ferrari Ascari*→ *car*' are generalized into '*Ferrari*→ *car*'.

ing their argument mapping. It is based on WordNet and NomLex-plus (Meyers et al., 2004), verified statistically through intersection with the unary-DIRT algorithm (Szpektor & Dagan, 2008). AmWN rules are defined between unary templates, for example, *'kill X→X die'*

These automatically-extracted inference rules lack two attributes defined in our formalism: rule type (substitution/introduction) and explicit alignments (beyond alignments between R's variables and their L counterparts, which are defined by default). These attributes are added automatically using the following heuristics:

1. If the roots of L and R have the same part-of-speech, then it is a substitution rule (e.g., '*X buy Y→ Y was sold to X*'). Otherwise (e.g., '*Y's acquisition by X→ Y was sold to X*'), it is an introduction rule.

2. The roots of L and R are assumed to be aligned.

Note that application of some of the above rules, (e.g., WordNet derivations and some of the rules learned by DIRT), does not result in a valid parse tree. These rules should not be used when aiming for exact derivation of h from t. However, they may be useful when the inference engine is used together with an approximate matching component, as in our RTE system. Our approximate matcher (described in Section 7.3) employs features such as the coverage of words and subtrees in h by $\mathcal{F}$, and therefore can benefit from such inferences. These rules should preferably be applied only as the last step of the inference process, to avoid cascading errors.

7. Evaluation

In this section, we present an empirical evaluation of our entailment system as a whole, as well as evaluation of its individual components. We evaluate both the quality of the system's output (in terms of accuracy, precision, and recall) and its computational efficiency (in terms of running time and space, using various application settings.

We first evaluate the knowledge-based inference engine. In Section 7.1, we describe an experiment in which the engine aims to prove simple template hypotheses, representing binary predicates, from texts sampled from a large corpus. Next, in Section 7.2 we evaluate the efficiency of our engine implementation using the compact forest data structure. We then evaluate the complete entailment system, including the approximate entailment classifier (Section 7.3). Finally, in Sections 7.4–7.5 we provide an in-depth analysis of the performance of the inference component on RTE data.

7.1 Proof System Evaluation

In this experiment, we evaluate the inference engine on finding strict proofs. That is, the inference process must derive precisely the target hypothesis (or an instantiation of it, in the case of template hypotheses, which contain variables as defined in Section 4.7). Thus, we should evaluate its precision over text-hypothesis pairs for which a complete proof chain is found, using the available rules. We note that the PASCAL RTE datasets are not suitable for this purpose. These rather small datasets include many text-hypothesis pairs for

which available inference rules would not suffice for deriving complete proofs. Furthermore, since the focus of this research is applied textual inference, the inference engine should be evaluated in an NLP application setting where the texts represent realistic distribution of linguistic phenomena. Manually-composed benchmarks such as the FraCas test suite (Cooper et al., 1996), which contains synthetic examples for specific semantic phenomena, are clearly not suitable for such an evaluation.

As an alternative, we chose a Relation Extraction (RE) setting, for which complete proofs can be achieved for a large number of corpus sentences. In this setting, the system needs to identify pairs of arguments in sentences for a target semantic relation (e.g., *'X buy Y'*).

7.1.1 System Configuration

In this experiment, which was first reported by Bar-Haim et al. (2007), we used an earlier version of the engine and the rule bases. The engine in this experiment does not make use of the compact forest, but rather generates each consequent explicitly. Polarity annotations are not propagated from source to derived trees. Instead, polarity annotation rules are applied to the original text t, and to each inferred consequent, prior to application of any inference rule. The following rule bases were used in this experiment:

Generic Linguistic Rules We used the generic rule base presented in Section 6, including both inference and the polarity annotation rules. This early version did not include the lexicalized polarity rules derived from VerbNet and from the PARC lexicon (category 5 in Table 3).

Lexical-Syntactic Rules *Nominalization rules:* inference rules such as 'X's acquisition of $Y \rightarrow X$ acquired Y' capture the relations between verbs and their nominalizations. These rules were derived automatically (Ron, 2006) from Nomlex, a hand-coded database of English nominalizations (Macleod, Grishman, Meyers, Barrett, & Reeves, 1998), and from WordNet.

Automatically Learned Rules: we used the DIRT paraphrase collection, as well the output of TEASE (Szpektor et al., 2004), another unsupervised algorithm for learning lexical-syntactic rules. TEASE acquires entailment relations from the Web for a given input template I by identifying characteristic variable instantiations shared by I and other templates. Both algorithms provide a ranked list of output templates for a given input template. Some of the learned rules are linguistic paraphrases, (e.g., 'X confirm $Y \rightarrow X$ approve Y'), while others capture world knowledge, (e.g., 'X buy $Y \rightarrow X$ own Y'). These algorithms do not learn the entailment direction of the rule, which reduces their accuracy when applied in any given direction. For each system, we considered the top 15 bi-directional rules learned for each template.

Generic Default Rules These rules are used to define default behavior, in situations where no case-by-case rules are available. We used one default rule that allows removal of any modifiers from nodes. Ideally, this rule would be replaced in future work by more specific rules for removing modifiers.

7.1.2 Evaluation Process

We use a sample of test template hypotheses that correspond to typical RE relations, such as *'X approve Y'*. We then identify in a large test corpus, sentences from which an instantiation of the test hypothesis is proved. For example, the sentence *"the budget was approved by the parliament"* is found to prove the instantiated hypothesis *"parliament approve budget"* (via the passive-to-active inference rule). Finally, a sample of such candidate sentences-hypothesis pairs is judged manually for true entailment. We repeated the process to compare different system configurations.

Since the publicly available sample output of TEASE is much smaller than the other resources[12] we randomly selected from this resource 9 transitive verbs that may correspond to typical RE predicates[13]. We formed test templates by adding subject and object variable nodes. For example, for the verb *accuse* we constructed the template '$X_{\text{NOUN}} \xleftarrow{subj} accuse_{\text{VERB}} \xrightarrow{obj} Y_{\text{NOUN}}$'.

For each test template h we identify sentences in the corpus from which the template can be proved by our system. To efficiently find proof chains that generate h from corpus sentences we combine forward and backward (Breadth-First) searches over the available rules. First, we use a backward search over the lexical-syntactic rules, starting with rules whose right-hand-side is identical to the test template. The process of backward chaining the DIRT/TEASE and nominalization rules generates a set of templates t_i, all of them proving (deriving) h. For example, for the hypothesis *'X approve Y'* we may generate the template *'X confirm Y'*, through backward application of a DIRT/TEASE rule, and then further generate the template *'confirmation of Y by X'*, through a nominalization rule. Since the templates t_i are generated by lexical-syntactic rules, which modify open-class lexical items, they may be considered "lexical expansions" of h.

Next, for each specific t_i we generate a search engine query composed of the open-class words in t_i. This query fetches candidate sentences from the corpus, from which t_i might be proven using the generic linguistic rules (recall that these rules do not modify open-class words). To that end, we use a forward search that applies the generic rules, starting from a candidate sentence s and trying to derive t_i by a sequence of rule applications. If successful, the variables in t_i are instantiated (cf. Section 4.7). Consequently, we know that under these variable instantiations, h can be proven from s (since s derives t_i which in turn derives h).

We performed the above search for sentences that prove each test template over the Reuters RCV1 corpus, CD#2, applying Minipar for parsing. Through random sampling, we obtained 30 sentences that prove (according to the tested system configuration) each of the 9 test templates, yielding a total of 270 pairs of a sentence, and an instantiated hypothesis, for each of the four tested configurations, described below (1080 pairs overall). These pairs were split for entailment judgment between two human annotators (graduate students at the Bar-Ilan NLP group). The annotators achieved, on a sample of 100 shared exam-

12. The output of TEASE and DIRT, as well as many other knowledge resources, is available from the RTE knowledge resources page:
`http://aclweb.org/aclwiki/index.php?title=RTE_Knowledge_Resources`
13. The verbs are *approach, approve, consult, lead, observe, play, seek, sign, strike.*

#	Configuration	Precision	Yield
1	Baseline (embed h anywhere in s)	67.0%	2,414
2	Proof (embed h at the root of s)	78.5%	1,426
3	Proof + Generic	74.8%	2,967
4	Proof + Generic + Lexical-Syntactic	23.6%	18,809

Table 4: Proof system evaluation

ples, an agreement level of 87%, and a Kappa value of 0.71 (corresponding to "substantial agreement").

7.1.3 Results

We tested four configurations of the proof system:

1. Baseline: The baseline configuration follows the prominent approach in graph-based entailment systems: the system tries to embed the given hypothesis anywhere in the candidate sentence tree s, while only negative or unknown polarity (detected by the annotation rules) may block embedding.

2. Proof: In this configuration h has to be strictly generated from the candidate sentence s. The only inference rule available is the default rule for removing modifiers (polarity annotation rules are active as in Baseline). This configuration is equivalent to embedding h in s with the root of h matched at the root of s, since modifiers that are not part of the match can be removed from s by the default rule. However, if h is embedded elsewhere in s it will not be extracted, as opposed to the Baseline configuration.

3. Proof + Generic: As Proof, plus generic linguistic rules.

4. Proof + Generic + Lexical-Syntactic: As the previous configuration, plus lexical-syntactic rules.

For each system configuration we measure *precision*, the percentage of examples judged as correct (entailing), and average *extrapolated yield*, which is the expected number of truly entailing sentences in the corpus that would be proven as such by the system. The extrapolated yield for a specific template is calculated as the number of sample sentences judged as entailing, multiplied by the sampling proportion. The average is calculated over all test templates. We note that, similar to IR evaluations, it is not possible to compute true recall in our setting since the total number of entailing sentences in the corpus is not known (recall is equal to the yield divided by this total). However, it is straightforward to measure *relative* recall differences among different configurations based on the yield. Thus, using these two measures estimated from a large corpus it is possible to conduct robust comparison between different configurations, and reliably estimate the impact of different rule types. Such analysis is not possible with the RTE datasets, which are rather small, and their hand-picked examples do not represent the actual distribution of linguistic phenomena.

The results are reported in Table 4. First, comparing the results for Proof with the results for Baseline, we observe that the requirement for matching h at the root of s (i.e., at the main clause of s), rather than allowing it to be matched anywhere in s, improves the precision considerably over the baseline (by 11.5%), while reducing the yield by nearly 40%. The Proof configuration avoids errors resulting from improper extraction of embedded clauses.

Remarkably, using the generic inference rules, our system is able to gain back the lost yield in Proof and further surpass the yield of the baseline configuration. In addition, we obtain a higher precision than the baseline (a 7.8% difference), which is statistically significant at a $p < 0.05$ level, using z test for proportions. This demonstrates that our principled proof approach appears to be superior to the more heuristic baseline embedding approach, and exemplifies the contribution of our generic rule base. Overall, generic rules were used in 46% of the proofs.

Adding the lexical-syntactic rules increased the yield by a factor of six. This shows the importance of acquiring lexical-syntactic variability patterns. However, the precision of DIRT and TEASE is currently quite low, causing overall low precision. Manual filtering of rules learned by these systems is currently required to obtain reasonable precision.

Error analysis revealed that for the third configuration Proof + Generic rules, a significant 65% of the errors are due to parsing errors, most notably incorrect dependency relation assignment, incorrect POS assignment, incorrect argument selection, incorrect analysis of complex verbs (e.g., *play down* in the text vs. *play* in the hypothesis) and ungrammatical sentence fragments. Another 30% of the errors represent conditionals, negation, and modality phenomena, most of which could be handled by additional rules, some making use of more elaborate syntactic information such as verb tense. The remaining, and rather small, 5% of the errors represent truly ambiguous sentences which would require considerable world knowledge for successful analysis.

7.2 Compact Forest Efficiency Evaluation

Next, we evaluate the efficiency of compact inference (cf. Section 5) in the setting of recognizing textual entailment, using the RTE-3 and RTE-4 datasets (Giampiccolo et al., 2007, 2008). These datasets consist of (*text, hypothesis*) pairs, which need to be classified as *entailing/non entailing*. Our first experiment, using the generic inference rule set, shows that compact inference outperforms explicit inference (efficiency-wise) by orders of magnitude (Section 7.2.1). The second experiment shows that compact inference scales well to a full-blown RTE setting with several large-scale rule bases, where up to hundreds of rules are applied per text (Section 7.2.2).

7.2.1 Compact vs. Explicit Inference

To compare explicit and compact inference we randomly sampled 100 pairs from the RTE-3 development set, and parsed the *text* in each pair using Minipar (Lin, 1998). To avoid memory overflow for explicit inference, we applied to these sentences only the subset of generic inference rules described in Section 6.1. For a fair comparison, we aimed to make the explicit inference implementation reasonably efficient, for example by preventing multiple generations of the same tree by different permutations of the same rule applications. Both

	Compact	Explicit	Ratio
Time (msec)	61	24,184	396
Rule applications	12	123	10
Node count	69	5,901	86
Edge endpoints	141	11,552	82

Table 5: Compact vs. explicit inference, using generic rules. Results are averaged per text-hypothesis pair.

configurations perform rule application iteratively, until no new matches are found. In each iteration, we first find all rule matches and then apply all matching rules. We compare run time, number of rule applications, and the overall generated size of nodes and edges, where edge size is represented by the sum of its endpoints (2 for a regular edge, $|S_d| + |T_d|$ for a d-edge).

The results are summarized in Table 5. As expected, the results show that compact inference is by orders of magnitude more efficient than explicit inference. To avoid memory overflow, inference was terminated after reaching 100,000 nodes. Three out of the 100 test texts reached that limit with explicit inference, while the maximal node count for compact inference was only 268. The number of rule applications is reduced due to the sharing of common subtrees in the compact forest, by which a single rule application operates simultaneously over a large number of embedded trees. The results suggest that scaling to larger rule bases and longer inference chains would be feasible for compact inference, but prohibitive for explicit inference.

7.2.2 Application to an RTE System

The goal of the second experiment was to test if compact inference scales well for broad inference rule bases. In this experiment we used the Bar-Ilan RTE system (Bar-Haim et al., 2008). The system operates in two primary stages:

Inference: inference rules are first applied to the initial compact forest $\mathcal{F}$, aiming to bring it closer to the hypothesis h. In this experiment, we use all the knowledge bases described in Section 6. Overall, these rule bases contain millions of rules.

In the current system we implemented a simple search strategy, in the spirit of (de Salvo Braz et al., 2005): first, we applied three exhaustive iterations of generic rules. Since these rules have low fan-out (few possible right-hand-sides for a given left-hand-side), it is affordable to apply and chain them more freely. At each iteration we first find all rule matches, and then apply all matched rules. To avoid repeated identical rule applications, we mark newly added nodes at each iteration, and in the next iteration consider only matches containing new nodes. We then perform a single iteration of all other lexical and lexical-syntactic rules, applying them only if their L part was matched in $\mathcal{F}$ and their R part was matched in h. Further investigation of effective search heuristics over our representation is left for future research.

Classification: Following inference, a set of features is extracted from the resulting $\mathcal{F}$ and from h and fed into an SVM classifier, which determines entailment. We describe the

	RTE3-Dev		RTE4	
	Avg.	Max.	Avg.	Max.
Rule applications	14	275	15	110
Node count	71	606	80	357
Edge endpoints	155	1,741	173	1,062

Table 6: Application of compact inference to the RTE-3 Dev. and RTE-4 datasets, using all rule types

classification stage in more detail in the next section, which discusses the performance of our RTE system.

Table 6 provides statistics on rule applications using all rule bases, over the RTE-3 development set and the RTE-4 dataset[14]. Overall, the primary result is that the compact forest indeed accommodates well extensive rule applications from large-scale rule bases. The resulting forest size is kept small, even in the maximal cases which were causing memory overflow for explicit inference.

7.3 Complete RTE System Evaluation

In the previous sections, we evaluated our knowledge-based inference engine (the proof system) with respect to the quality of its output (precision, recall) as well as its computational efficiency (time, space). We now evaluate the complete RTE system, which combines the inference engine with an approximate classification module.

The classification setting and its features are quite typical for the RTE literature. Features can be broadly categorized into two subsets: (a) lexical features that solely depend on the lexical items in $\mathcal{F}$ and h, and (b) lexical-syntactic features that also take into account the syntactic structures and dependency relations in $\mathcal{F}$ and h. Below is a brief description of the features. A complete description appears in our RTE system report (Bar-Haim et al., 2008).

Lexical features: *Coverage features* check if the words in h are present (covered) in $\mathcal{F}$. We assume that a high degree of lexical coverage correlates with entailment. These features measure the proportion of uncovered content words, verbs, nouns, adjectives and adverbs, named entities and numbers. *Polarity mismatch* features detect cases where nouns or verbs in h are only matched in $\mathcal{F}$ with incompatible polarity. These features are assumed to indicate non-entailment.

Edge coverage features: We say that an edge in h is *matched* in $\mathcal{F}$ if there is an edge in $\mathcal{F}$ with matching *relation*, *source* node and *target* node. We say an edge in h is *loosely-matched* if there is some path in $\mathcal{F}$ from a matching *source* node to a matching *target* node. Based on these definitions we extract two features: the proportion of h edges *matched/loosely matched* in $\mathcal{F}$.[15]

14. Running time is not included since most of it was dedicated to rule fetching, which was rather slow for our available implementation of some resources. The elapsed time was a few seconds per (t, h) pair.

15. We only look at a subset of the edges labeled with relevant dependency relations.

Predicate-argument features: If $\mathcal{F}$ entails h, then the predicates in h should be matched in $\mathcal{F}$ along with their arguments. Predicates include verbs (except for the verb "be") or subject complements in copular sentences, For example, *smart* in *Joseph is smart*. Arguments are the daughters of the predicate node in h.[16] Four features are computed for each $\mathcal{F}, h$ pair. We categorize every predicate in h that has a match in $\mathcal{F}$ to one or more of four possible categories:

1. *complete match* - a matching predicate exists in $\mathcal{F}$ with matching arguments and dependency relations.
2. *partial match* - a matching predicate exists in $\mathcal{F}$ with some matching arguments and dependency relations.
3. *opposite match* - a matching predicate exists in $\mathcal{F}$ with some matching arguments but incorrect dependency relations.
4. *no match* - no matching predicate in $\mathcal{F}$ has any matching arguments.

If a predicate is categorized as a *complete match* it will not be in any other category. Finally, we compute the four features for the $\mathcal{F}, h$ pair: the proportion of predicates in h that have a *complete match* in $\mathcal{F}$, and three binary features, checking if there is any predicate in h categorized as a *partial match/opposite match/no match*. Since the *subject* and *object* arguments are crucial for textual entailment, we compute four similar features only for the subset of predicates that have these arguments (ignoring other arguments).

A global lexical-syntactic feature: This feature measures how well the subtrees in h are covered by $\mathcal{F}$, weighted according to the proximity to the root of h. This feature is somewhat similar to the dependency tree kernel of Collins and Duffy (2001), which measures the similarity between two dependency trees by counting their common subtrees. However, our measure has several distinct properties which makes it suitable for our needs: (a) It's a directional measure, estimating the coverage of h by $\mathcal{F}$, but not vice versa (b) It operates on a compact forest and a tree, rather than on a pair of trees. (c) It takes into account the distance from the root of h, assuming that nodes closer to the root are more important.

The system was trained on the RTE-3 development set, and was tested on the RTE3 and RTE-4 test sets (no development set was released for RTE-4). Co-reference substitution was disabled due to the insufficient accuracy of the co-reference resolution tool we used. We first report its overall performance, and then provide some analysis of the inference module, which is our focus in this work.

The accuracies obtained in this experiment are shown in Table 7 (under the "inference" column). The results on RTE-3 are quite competitive: compared to our 66.4%, only 3 teams out of the 26 who participated in RTE-3 scored higher than 67%, and three more systems scored between 66% and 67%. The results for RTE4 rank 9-10 out of 26, with only 6 teams scoring higher by more than 1%. Overall, these results show that our system is well-situated in the state of the art for the RTE task.

Table 8 provides a more detailed view of our system's performance. Precision, recall, and F1 results are given for both entailing and non-entailing pairs, as well as the overall accuracy.

16. When the dependent is a preposition or a clause we take the complement of the preposition or the head of the clause respectively as the dependent.

The table also shows the results per task (IE, IR, QA and SUM). Overall, our system tends to predict entailment more often than non-entailment. The recall for entailing pairs is much higher than the recall for non-entailing pairs, while the precision for non-entailing pairs is much higher than for entailing pairs. Performance varies considerably among different tasks. Our RTE3 accuracy results for QA and IR are considerably higher than the average results achieved by RTE3 submissions, as reported by the organizers (Giampiccolo et al., 2007) (0.71 and 0.66, respectively), while for IE and SUM, our results are a bit above the average (0.52 and 0.58). Our RTE4 results are better for IR and SUM, which seem to be the easier tasks in RTE4 (Giampiccolo et al., 2008).[17]

7.4 Usage and Contribution of Knowledge Bases

To evaluate the accuracy gain from knowledge-based inference, we ran the system with the inference module disabled, so that entailment classification is applied directly to the initial parse tree of the text. The results are shown under the "no inference" column of Table 7. Comparing these results to the full system accuracy ("inference"), we see that applying the inference module resulted in higher accuracy on both test sets. The contribution was more prominent for the RTE-4 dataset. These results illustrate a typical contribution of current knowledge sources for current RTE systems. This contribution is likely to increase with current and near future research, on topics such as extending and improving knowledge resources, applying them only in semantically suitable contexts, improved classification features, and broader search strategies.

Tables 9 and 10 illustrate the usage and contribution of individual rule bases. Table 9 shows the distribution of rule applications over the various rule bases. Table 10 presents ablation study showing the marginal accuracy gain for each rule base. These results show that each of the rule bases is applicable for a large portion of the pairs, and contributes to the overall accuracy. We note that the results are highly dependent on the search strategy. For instance, chaining of lexical rules is expected to increase the number of lexical rule applications, but reduce their accuracy. We provide a more detailed analysis of rule applications in our system in the next section.

7.5 Manual Analysis

We conclude the evaluation with two manual analyses of the inference component within the RTE system. The first analysis (Subsection 7.5.1) assesses the applicability of our inference framework to the RTE task as well as the actual coverage of the current system. It also categorizes the cases in which our formalism falls short. We then (Subsection 7.5.2) assess the correctness of the applied rules, and analyze the various causes for incorrect applications. The analyses were done by one of the authors on randomly sampled subsets of the RTE-3 test set.

17. According to the RTE4 organizers, the IE task appeared to be the most difficult task, while SUM and IR seemed to be the easier tasks. However, they did not report the average accuracy per task.

Test set	Accuracy		Δ	Lexical Overlap	Best RTE Result
	No inference	Inference			
RTE3	64.6%	66.4%	1.8 %	62.4%	80.0%
RTE4	57.5%	60.6%	*3.1%	56.6%	74.6%

Table 7: Inference contribution to RTE performance. The system was trained on the RTE-3 development set. * indicates statistically significant difference (at level $p < 0.02$, using McNemar's test). The best results achieved in the RTE3 and RTE4 challenges (Hickl & Bensley, 2007; Bensley & Hickl, 2008), as well as lexical overlap baseline results (Mehdad & Magnini, 2009a), are also given for reference. Mehdad and Magnini have tested eight configurations of lexical overlap baselines, and chose the one that performs best on average over the RTE1-4 test sets.

		Non-Entailing Pairs			Entailing Pairs			
	Task	Precision	Recall	F1	Precision	Recall	F1	Accuracy
RTE3	IE	0.500	0.095	0.159	0.527	0.914	0.669	0.525
	IR	0.764	0.743	0.753	0.678	0.701	0.689	0.725
	QA	0.822	0.787	0.804	0.818	0.849	0.833	0.820
	SUM	0.545	0.341	0.420	0.600	0.777	0.677	0.585
	All	0.722	0.505	0.594	0.634	0.815	0.713	0.664
RTE4	IE	0.596	0.187	0.284	0.518	0.873	0.650	0.530
	IR	0.721	0.587	0.647	0.652	0.773	0.707	0.680
	QA	0.636	0.210	0.316	0.527	0.880	0.659	0.545
	SUM	0.685	0.630	0.656	0.657	0.710	0.683	0.670
	All	0.680	0.400	0.504	0.575	0.812	0.673	0.606

Table 8: RTE results breakdown by task and pair type

Rule base	RTE3-Dev		RTE4	
	Rules	App	Rules	App
WordNet	0.6	1.2	0.6	1.1
AmWN	0.3	0.4	0.3	0.4
Wikipedia	0.6	1.7	0.6	1.3
DIRT	0.5	0.7	0.5	1.0
Generic	4.7	10.4	5.4	11.5
Polarity	0.2	0.2	0.2	0.2

Table 9: Average number of rule applications per (t, h) pair, for each rule base. *App* counts each rule application, while *Rules* ignores multiple matches of the same rule in the same iteration.

7.5.1 Applicability and Coverage

This analysis assesses the ability of the inference framework to derive complete proofs for RTE (t,h) pairs in an idealized setting where perfect knowledge bases and co-reference resolution are available. This provides an upper bound to the coverage of our inference

Rule base	ΔAccuracy (RTE4)
WordNet	0.8%
AmWN	0.7%
Wikipedia	1.0%
DIRT	0.9%
Generic	0.4%
Polarity	0.9%

Table 10: Contribution of various rule bases. Results show accuracy loss on RTE-4, obtained when removing each rule base (ablation tests).

engine. A similar analysis was previously done by Bar-Haim, Szpektor, and Glickman (2005) on a subset of the RTE-1 dataset. However, here we go further and (a) assess the actual coverage of the required inferences by the implemented RTE system, and (b) present classification of uncovered cases into different categories.

We carried out the analysis as follows: 80 positive (entailing) pairs were randomly sampled from the RTE-3 test set. For each pair we aimed to manually derive a proof comprising inference steps that are expressible in our formalism, similar to the example in Section 2.2. If a complete proof could be derived, the pair was classified as *inferable*. Otherwise, it was classified into one of the following categories:

Discourse references: Complete proof requires incorporating pieces of information from the discourse, including event co-reference and bridging (Mirkin et al., 2010). Nominal co-reference substitution was not included, as it is covered in our formalism. For instance, in the text *"The Titanic's sinking after hitting an iceberg on April 14, 1912..."*, the year 1912 is not explicitly specified as the time of the Titanic's sinking, and this relation should be derived from the discourse in order to infer the hypothesis *"The Titanic sank in 1912"*.

Non-decomposable: The inference cannot be reasonably decomposed to a sequence of local rewrites. This is the case, for example, with the text *"The black plague lasted four years and killed about one-third of the population of Europe, or approximately 20 million people"* and the hypothesis *"Black plague swept Europe"*.

Other: A few other cases that did not fall into the above categories.

The distribution of these categories is shown in Table 11. We found that 60% of the pairs could be proven by our formalism given appropriate inference rules and co-reference information, which demonstrates the utility of our approach. The results are somewhat higher than the 50% reported by Bar-Haim et al. (2005), which may be attributed to the fact that RTE1 is considered a more difficult dataset, and entailment systems consistently perform better on RTE3.

Out of the remaining 40% pairs, our analysis highlights the significance of discourse references, which occur in 16.3% of the pairs. While previous analysis of discourse references in textual entailment was applied to the RTE-5 search task, where the text sentences are interpreted in the context of their full discourse (Mirkin et al., 2010), our analysis shows

Category	Count	%
Inferable	48	60.0%
Non-decomposable	14	17.5%
Discourse references	13	16.3%
Other	5	6.3%

Table 11: Applicability of our inference framework to the RTE task. 80 randomly selected entailing pairs from the RTE-3 test set were analyzed.

the significance of discourse references even for short, self-contained texts, of which RTE-3 is composed. Mirkin et al. show how our framework, and similar methods based on tree transformations, can be extended to utilize discourse references. Several works over the last few years targeted implied predicate-argument relationships, the most notable of which is the SemEval-2010 Task "Linking Events and Their Participants in Discourse" (Ruppenhofer, Sporleder, Morante, Baker, & Palmer, 2009). In particular, Stern and Dagan (2014) recently showed that identifying such relations improves the performance of their RTE system. Finally, the entailment in 17.5% of the pairs could not be established by a sequence of local rewrites, thus these cases are likely to require deeper methods for semantic analysis and inference.

The manually-derived proofs for the 48 inferable pairs included a total of 79 rule applications, an average of 1.65 rule applications per pair.[18] The maximal number of rules per pair was 3. 28 of these rules (35.4%) were applied in our system. 21% of the proofs for the inferable pairs were fully derived by our RTE system. Partial proofs were derived for additional 25% of the pairs. For the remaining 54% of the pairs, our system did not apply any of the rules in the manual proof. The results demonstrate the utility of the inference mechanisms and rule bases in our system, but on the other hand suggest that there is still much room for improvement in the coverage of the existing rule bases.

7.5.2 Correctness of Applied Rules

We next assess the correctness of rules applied by the inference engine. We focus on the four lexical and lexical-syntactic rule bases described in Section 6.3: WordNet, Wikipedia, DIRT, and Argument-Mapped WordNet (AmWN). Except for WordNet, these rule bases were generated automatically, therefore their accuracy is more of an issue than the accuracy of the manually-composed generic inference rules and polarity annotation rules. Furthermore, lexicalized rules are often context sensitive, which is an additional potential source of incorrect rule applications.

For this evaluation we randomly sampled 75 pairs from the RTE-3 test set, and analyzed all lexical and lexical-syntactic rule applications performed by the system for these pairs, a total of 201 rule applications. We define two levels of rule application correctness:

18. As previously mentioned, the RTE system does not apply rules that merely extract a subtree from a given source tree. Accordingly, such rules were ignored in this analysis as well.

Propositional: The derived tree resulting from the rule application is both grammatical and entailed from the source tree. This is the level of correctness assumed by our formalism.

Referential: In case the propositional correctness does not hold, we turn to the weaker criterion of *Referential Correctness*, following the notion of *Lexical Reference* (Glickman, Shnarch, & Dagan, 2006; Shnarch et al., 2009), which we extend here to the case of template-based rules with variables. Let rule $E : L \rightarrow R$ be an inference rule matched in a source tree s. Let l and r be the instantiations of L and R respectively, according to the variable matching of L in s. We say that *referential correctness* holds if l generates a reference in s to a possible meaning of r. Some examples for such rules found in our analyzed sample are: *pope→papal*, *Turkish→Turkey* and *fishermen→fishing*. While these rule applications do not result in a valid entailed tree, they are still useful in the context of an RTE system that applies approximate matching (as previously discussed at the end of Section 6).

Incorrect rule applications were classified into one of the following categories:

1. *Bad rule:* The rule is a-priori incorrect (e.g., *Wales→year*).

2. *Bad context:* The rule is incorrect in the context of the source sentence. For example, the WordNet rule *strike→create* corresponds to the rare sense of *strike* defined as "produce by ignition or a blow" (as in "strike fire from the flint stone").

3. *Bad match:* The rule was applied due to incorrect matching of the left-hand-side, resulting from incorrect parse of the source tree.

The results are summarized in Table 12. Overall, 52.7% of the rule applications are correct. Interestingly, there are more referential (29.4%) than propositional (23.4%) rule applications. Unsurprisingly, the most accurate knowledge resource is the manually composed WordNet (75.9% correct applications), followed by the AmWN (57.9%) and Wikipedia (57.4%) rule bases, which were derived automatically from human-generated resources. The least accurate resource is DIRT (21.4%), which makes no use of human knowledge engineering, but rather was learned automatically based on corpus statistics. The accuracy of DIRT is considerably lower than the accuracy of the other resources, substantially decreasing the overall accuracy as well. Most of the errors for DIRT and Wikipedia are due to bad rules. This is also the overall dominant cause for incorrect applications, while for WordNet and AmWN the a-priori rule quality is very high and most of the errors are due to bad context. Wikipedia rules did not suffer from bad context, which can be explained by the fact that their left-hand-side was often an unambiguous named entity (*Madrid, Antelope Valley Freeway, Microsoft Office*). The analysis highlights the need for improving the accuracy of automatically-generated rule bases, whose quality is still far below human generated resources. The analysis also shows that context-sensitivity of lexicalized rules is still an issue even when these rules are applied conservatively as in our experiment (no chaining, both L and R were matched in $\mathcal{F}$ and h). This should be addressed in future research.

	DIRT	AmWN	Wikipedia	WordNet	All
% out of rule applications	27.9%	9.5%	33.8%	28.9%	100.0%
Propositional	17.9%	21.1%	19.1%	34.5%	23.4%
Referential	3.6%	36.8%	38.2%	41.4%	29.4%
Correct	21.4%	57.9%	57.4%	75.9%	52.7%
Bad rule	58.9%	5.3%	42.6%	0.0%	31.3%
Bad context	7.1%	31.6%	0.0%	17.2%	10.0%
Bad matching	12.5%	5.3%	0.0%	6.9%	6.0%
Incorrect	78.6%	42.1%	42.6%	24.1%	47.3%

Table 12: Analysis of lexical and lexical-syntactic rule applications

8. Discussion: Comparison to Related Approaches

In this section, we compare our work to several closely-related inference methods, most of which were described in Section 2.3.2.

The *discourse commitments* derived by Hickl (2008) are quite similar to the kind of consequents we generate by applying our syntactic, lexical-syntactic, and co-reference substitution rules. However, our work differs from Hickl's in several respects. First and foremost, Hickl's work does not fully describe a knowledge representation and inference framework, which is the main focus of our work. Hickl briefly mentions that the commitments were generated using a probabilistic FST-based extraction framework, but no further explanations or examples are given in the paper. Second, our framework allows unified modeling of a variety of inference types that are addressed by various tools and components in Hickl's system (FST, relation extraction, paraphrase acquisition, etc.). In addition, our system operates over lexical-syntactic representations, and does not rely on semantic parsing. Finally, the consequents generated in our formalism are packed in an efficient data structure, whereas Hickl's commitments are generated explicitly and he does not discuss commitment generation efficiency. It should be noted, however, that while explicit generation of commitments restricts the search space, it may simplify approximate matching (e.g., finding alignment between h and a given consequent vs. aligning h with the whole compact forest).

De Salvo Braz et al. (2005) presented a semantic inference framework that "augments" the text representation with only the right-hand-side of an applied rule, and in this respect is similar to ours. However, in their work, both rule application and the semantics of the resulting "augmented" structure were not fully specified. In particular, the distinction between individual consequents was lost in the augmented graph. By contrast, our compact inference is fully formalized and is proved to be equivalent to an expressive, well-defined formalism operating over individual trees, where each inferred consequent can be recovered from the compact forest.

MacCartney and Manning (2009) proposed a model of natural language inference which, similar to our framework, operates directly on parse-based representations. Their work extends previous work on *natural logic* (Valencia, 1991), which focused on semantic containment and monotonicity, by incorporating semantic exclusion and implicativity. They model the inference of h from t as a sequence of atomic edits; each can be thought of as generating an intermediate premise. Their calculus computes the semantic relation between the source

and the derived premise by propagating the semantic relation from the local edit upward through the parse tree according to the properties of intermediate nodes. For example, it can correctly infer that "*Some first-year students arrived* ⇒ *Some students arrived*", but "*Every first-year student arrived* ⇐ *Every student arrived*". The composition of these semantic relations along the inference chain yields the semantic relation holding between t and h. Their contribution is complementary to ours. In both approaches, the inference of h from t is modeled as a sequence of atomic steps (*rule applications* or *edits*). The focus of our framework is the representation and application of diverse types of transformations needed for textual inference, as well as efficient representation of possible inference chains. Application of an inference rule is assumed to always generate an entailed consequent, and polarity rules may be used to detect situations where this assumption does not hold and block rule application. By comparison, the formalism of MacCartney and Manning assumes rather simple edit operations, and is focused on precise predication of the semantic relation between t and h for a given sequence of edits that transform t into h. Thus, combining these two complementary approaches is a natural direction for future research.

9. Conclusion

The subject of this work was the representation and use of semantic knowledge for textual inference at the lexical-syntactic level. We defined a novel inference framework over parse trees, which represents diverse semantic knowledge as inference rules. The proof process aims to transform the source text into the target hypothesis through a sequence of rule applications, each generating an intermediate parse tree. A complementary contribution of this work is a novel data structure and an associated rule application algorithm, which are proved to be a valid implementation of the inference formalism. We illustrated inference efficiency both analytically and empirically.

Our approach has several advantageous properties. First, the ability to represent and apply a wide variety of inferences and combine them through rule chaining makes our framework more expressive than most of the previous RTE architectures. Second, this expressive power is obtained by a well-formalized and compact framework, based on unified knowledge representation and inference mechanisms. Finally, as shown by our RTE experiments, the compact forest data structure allows our approach to scale well to practical settings that involve very large rule bases and hundreds of rule applications per text-hypothesis pair.

We demonstrated the utility of our approach in two different semantic tasks. Experiments with unsupervised relation extraction showed that our exact proofs outperform the more heuristic common practice of hypothesis embedding. We also achieved competitive results on RTE benchmarks, by adding a simple approximate matching module to our inference engine. The contribution of semantic knowledge was illustrated on both tasks.

Limitations and possible extensions for our formalism were discussed in Section 4.8. Manual analysis of the inference engine's performance on the relation extraction and RTE tasks further suggested promising directions for future research, as discussed in Subsections 7.1.3 and 7.5. Two additional major areas for further research are the approximate matching heuristics and the proof search strategy. Stern and Dagan (2011) and Stern, Stern, Dagan, and Felner (2012) extended our work to address these two aspects, respectively.

Acknowledgments

This article is based on the doctoral dissertation of the first author, which was completed under the guidance of the second author at Bar-Ilan University (Bar-Haim, 2010). This work was partially supported by Israel Science Foundation grants 1095/05 and 1112/08, the IST Programme of the European Community under the PASCAL Network of Excellence IST-2002-506778, the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886, the Israel Internet Association (ISOC-IL), grant 9022, and the FBK-irst/Bar-Ilan University collaboration. The third author is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship. The authors wish to thank Cleo Condoravdi for making the polarity lexicon developed at PARC available for this research. We are grateful to Eyal Shnarch for his help in implementing the experimental setup described in Section 7.1. We also thank Iddo Greental for his collaboration on developing the generic rule base. Finally, we would like to thank Dan Roth, Idan Szpektor, Yonatan Aumann, Marco Pennacchiotti, Marc Dymetman and the anonymous reviewers for their valuable feedback on this work.

Appendix A: Compact Forest Complete Proofs

In this section, we provide complete proofs for the correctness of the compact inference algorithm presented in Section 5. We start with a few definitions.

Definition Let $L \rightarrow R$ be a rule matched and applied to a compact forest $\mathcal{F}$. As in Section 5.2, let l be a subtree of some represented tree $t \in \mathcal{T}(\mathcal{F})$, in which L is matched. Recall that S_L was defined as l excluding nodes matched by dual-leaf variables, and similarly S_R was defined as a copy of R without its dual-leaf variables that is generated and inserted into $\mathcal{F}$ as part of rule application. The roots of S_L and S_R are denoted r_L and r_R respectively. We say that a node $s \in S_R$ is *tied to* a node $s' \in S_L$, if s is set as a source node for one of the outgoing d-edges of s', due to alignment sharing or dual leaf variable sharing.

The graph operations performed when applying a rule $L \rightarrow R$ to a compact forest $\mathcal{F}$ can be summarized as follows:

1. Adding the subtree S_R to $\mathcal{F}$.
2. Setting r_R as a target node of a d-edge in $\mathcal{F}$.
3. Setting nodes in S_R that are tied to nodes in S_L as source nodes for d-edges in $\mathcal{F}$, according to the rules of variable sharing and dual leaf variable sharing. Recall that these d-edges are not part of S_L.

First, we show a simple property of cDGs generated by the inference process:

Lemma 1 *Every node in a cDG generated by the inference process has at most one incoming d-edge.*

Proof By construction, in the initial forest each node has at most one incoming d-edge. Each rule application adds a subtree S_R, whose nodes have at most one incoming d-edge. Last, the root r_R, which initially has no incoming edges, is set as a target for a single d-edge during rule application (the incoming d-edge of r_L). Therefore, the lemma follows by induction on the number of rule applications. ∎

Using the following theorem we show that the inference process generates a compact forest:

Theorem 1 *Applying a rule to a compact forest results in a compact forest.*

Proof Let $\mathcal{F}'$ be the cDG generated by applying the rule $L \rightarrow R$ to a compact forest $\mathcal{F}$. We show that $\mathcal{F}'$ is a compact forest, that is, a cDAG with a single root r where all the embedded DAGs rooted in r are trees. First, we show that $\mathcal{F}'$ is a cDAG, (i.e., it does not contain a cycle of e-edges).

Assume by contradiction that $\mathcal{F}'$ contains a simple cycle of e-edges C. Applying the rule $L \rightarrow R$ did not add any e-edges between nodes in $\mathcal{F}$. Therefore, C must pass through r_R, the root of S_R and contain an e-edge (p, r_R). Since S_R is a tree, C must also leave S_R through an e-edge (u, v) ($u \in S_R$ and $v \notin S_R$). The cycle can be written as $p \rightarrow r_R \rightarrow ... \rightarrow u \rightarrow v \rightarrow ... \rightarrow p$. Notice that the path from v to p is fully contained in $\mathcal{F}$ since the cycle C is simple and entering S_R is possible only through r_R.

$L \rightarrow R$ must be a substitution rule, otherwise p would be the root of $\mathcal{F}$. This is impossible, since the root has no incoming d-edges. Therefore, r_R and r_L have the same single incoming d-edge, and the e-edge (p, r_L) exists in $\mathcal{F}$. In addition, u was added as a source node of a d-edge d in $\mathcal{F}$ since it is tied to some $u' \in S_L$, which is also a source node of d. Therefore, the path $r_L \rightarrow ... \rightarrow u' \rightarrow v$ exists in $\mathcal{F}$. Finally, we know that the path from v to p is fully contained in $\mathcal{F}$, therefore we can construct a cycle $p \rightarrow r_L \rightarrow ... \rightarrow u' \rightarrow v \rightarrow ... \rightarrow p$ in $\mathcal{F}$, in contradiction to our assumption that $\mathcal{F}$ is a compact forest.

We have shown that $\mathcal{F}'$ is a cDAG. Next, we define a generalization for embedded DAGs, which will help us show that all embedded DAGs in $\mathcal{F}'$ rooted in r are trees.

Definition An *embedded partial DAG* $G = (V, E)$ in a cDAG $\mathcal{G}$ rooted in a node $v \in \mathcal{V}$ is similar to an embedded DAG and is generated using the following process:

1. Initialize G with v alone
2. Repeat any number of iterations:
   (a) choose a node $s \in V$
   (b) choose an outgoing d-edge d of s that was not already chosen by s in a previous iteration. If all d-edges have been chosen - halt.
   (c) choose a target node $t \in T_d$ and add the e-edge $(s, t)_d$ to G.

We now show that all embedded partial DAGs in $\mathcal{F}'$ rooted in any node are trees. Since an embedded DAG is also an embedded partial DAG, this proves that all embedded DAGs in $\mathcal{F}'$ rooted in r are trees. Assume by contradiction that after applying $L \rightarrow R$ there is an

embedded partial DAG T' rooted at a node n that is not a tree. We can assume that n is not in S_R, otherwise, we can extend T' by adding a path $p \rightarrow r_R \rightarrow ... \rightarrow n$, where p is a node outside S_R that is a source node of the incoming edge of r_R.

Since T' is not a tree, there are two simple paths P_1 and P_2 from n that reach some node z from two different e-edges. z cannot be in S_R, since any two paths that meet in the subtree S_R, must first meet in its root r_R entering its incoming d-edge. However, we could then construct in $\mathcal{F}$ the same two paths, only selecting r_L instead of r_R, in contradiction with the assumption that $\mathcal{F}$ is a compact forest. Clearly, either P_1 or P_2 must pass through the new subtree S_R, otherwise the two paths already existed in $\mathcal{F}$.

We first handle the case where, without loss of generality, P_1 passes through S_R and P_2 does not. P_1 passes through S_R and contains an e-edge (p, r_R). Since $z \notin S_R$, it also contains an e-edge (u, v) such that $u \in S_R$ and $v \notin S_R$. So P_1 can be written as $n \rightarrow ... \rightarrow p \rightarrow r_R \rightarrow ... \rightarrow u \rightarrow v \rightarrow ... \rightarrow z$. The paths from n to p and from v to z are in $\mathcal{F}$, because the only way to enter S_R is through r_R and P_1 is simple. We can now incrementally construct in $\mathcal{F}$ the following embedded partial DAG T: First, we construct P_2 and the section in P_1 from n to p as in T'. Next, we expand p with the e-edge (p, r_L) instead of (p, r_R). We would like to expand T from r_L and reach z if possible.

As previously explained, u is tied to a node $u' \in S_L$ and therefore the e-edge (u', v) exists in $\mathcal{F}$. Therefore, there is a path P' in S_L, from r_L through (u', v) to z. However, it is not guaranteed that the whole P' can be added to T. We try to expand T incrementally with P', at each step adding the next e-edge in the path. If we succeed, then T is an embedded graph in $\mathcal{F}$ with two paths to z, a contradiction. If we fail, this can only be due to an e-edge $(z', t) \in P'$ we cannot add. Thus, z' must already be in P_2, and is a node for which there are two distinct paths in the embedded graph T, a contradiction. The path constructed is indeed different from P_2 since it contains the e-edge (p, r_L) that cannot be part of P_2, since P_1 contains the disjoint edge (p, r_R).

In the remaining case, both P_1 and P_2 pass through S_R and reach a node $z \notin S_R$. P_1 can be written as $n \rightarrow ... \rightarrow u_1 \rightarrow v_1 \rightarrow ... \rightarrow z$ and P_2 as $n \rightarrow ... \rightarrow u_2 \rightarrow v_2 \rightarrow ... \rightarrow z$, where u_1, $u_2 \in S_R$, and v_1, $v_2 \notin S_R$. Assume first that the e-edges (u_1, v_1) and (u_2, v_2) originate from the same d-edge d. Then $u_1 \neq u_2$, otherwise (u_1, v_1) and (u_2, v_2) could not be both in the same embedded partial DAG. u_1,u_2 are tied to the nodes u'_1, $u'_2 \in S_L$.

We show that $u'_1 \neq u'_2$: Assume by contradiction that $u'_1 = u'_2 = u'$. u' is tied to u_1 and u_2 due to alignment sharing or dual leaf variable sharing. u' cannot be tied to both u_1 and u_2 due to alignment sharing since alignment is a function from nodes in S_L to nodes in S_R. It cannot be tied to both due to dual leaf variable sharing, since any variable appears only once in R. Finally, if u' is tied to u_1 (without loss of generality) due to dual leaf variable sharing, then the d-edge d is part of l. Therefore, u_2 will not include d as an aligned modifier, and thus u_2 will not be tied to u' due to alignment.

We can now construct an embedded graph T rooted at r_L in $\mathcal{F}$: Because S_L is part of the match of L in $\mathcal{F}$, we can construct an embedded graph rooted at r_L with a path to any node in S_L, in particular with paths to u'_1 and u'_2. Since $u'_1 \neq u'_2$, and both u'_1 and u'_2 are source nodes of d, which is not part of S_L, we can expand these two paths with the e-edges (u'_1, v_1) and (u'_2, v_1) of d and get an embedded graph in $\mathcal{G}_n$ that is not a tree, a contradiction.

Suppose that the e-edges (u_1, v_1) and (u_2, v_2) originate from different d-edges d_1 and d_2 respectively. u_1 and u_2 are tied to u'_1 and u'_2. Therefore, if $v_1 \neq v_2$ we can construct the following embedded graph T rooted at r_L: as in the previous case, we can expand the paths in S_L from r_L to u'_1 and u'_2. Next, we add the e-edges (u'_1, v_1) of d_1 and (u'_2, v_2) of d_2. Recall that d_1 and d_2 are not in S_L and can therefore be used for expansion. We try to expand this embedded graph to include the paths from v_1 and v_2 to z. If we succeed, we have two paths in T leading to z. If we fail we have two paths in T_n meeting at some other node z', as explained above. Last, if $v_1 = v_2 = v$, then v is a node in $\mathcal{F}$ that has two incoming d-edges, contradicting Lemma 1.

The case of an introduction rule is quite similar but simpler. If P_1 passes through S_R and P_2 does not, then n must be the root of the compact forest (the only node with a path to r_R). However, in this case n has a single outgoing d-edge, and therefore all its outgoing e-edges are disjoint (i.e. cannot be part of the same embedded DAG). Thus, P_2 must also pass through r_R - a contradiction. If both P_1 and P_2 pass through S_R, the proof is identical to the case of a substitution rule.

We have shown that $\mathcal{F}'$ is a cDAG whose embedded DAGs rooted in r are trees. $\mathcal{F}'$ also has a single root because all new nodes added when applying $L \rightarrow R$ have an incoming edge. Hence, $\mathcal{F}'$ is a compact forest. ∎

Corollary 1 *The inference process generates a compact forest.*

Proof It is easy to verify that initialization generates a compact forest. Since applying a rule to a compact forest results in a compact forest, the inference process generates a compact forest by induction on the number of rule applications. ∎

Theorem 2 *Given a rule base $\mathcal{R}$ and a set of initial trees T, a tree t is represented by a compact forest derivable from T by the inference process $\Leftrightarrow$ t is a consequent of T according to the inference formalism.*

Proof ($\Leftarrow$) We first show completeness by induction on the number of rule applications n. If $n = 0$ then t is one of the initial trees and is represented by the initial compact forest. Let t_{n+1} be a tree derived in the formalism by applying a sequence of $n+1$ rules. We show that t_{n+1} is represented in a derivable compact forest. t_{n+1} was derived by applying the rule $L \rightarrow R$ to the tree t_n. According to the inductive assumption, t_n is represented in a compact forest $\mathcal{F}$ derivable by the inference process. Therefore, the rule $L \rightarrow R$ can be matched and applied in $\mathcal{F}$. We assume $L \rightarrow R$ is a substitution rule since the case of an introduction rule is similar. t_{n+1} is almost identical to t_n except it contains the subtree R instead of L with instantiated variables and aligned modifiers. It is easy to verify that after application of $L \rightarrow R$ on $\mathcal{F}$ resulting in $\mathcal{F}'$, $\mathcal{F}'$ will contain an embedded tree t that is almost identical to t_n, except that the root of S_R, r_R, will be chosen instead of the root of S_L, r_L, and the rest of S_R can also be chosen with the appropriate instantiated variables and modifiers. Therefore, $t_{n+1} = t$ is contained in $\mathcal{F}'$ as required. t is guaranteed to be a tree according to Corollary 1.

($\Rightarrow$) Next, we prove soundness by induction on the number of rule applications in the forest. At initialization, all of the initial trees are consequents. Let $\mathcal{F}_{n+1}$ be a compact forest derived by $n+1$ rule applications (Corollary 1 guarantees that $\mathcal{F}_{n+1}$ is indeed a

compact forest). Given a tree t_{n+1} represented by $\mathcal{F}_{n+1}$, we show that t_{n+1} is a consequent in the formalism.

If t_{n+1} was already represented by the compact forest after n rule applications, then according to the assumption of the induction it is a consequent in the formalism. If not, then t_{n+1} is a new embedded tree created after the application of the rule $L \rightarrow R$. Therefore, t_{n+1} contains the entire subtree S_R. We now incrementally construct an embedded tree t_n represented by $\mathcal{F}_n$ such that t_{n+1} is the result of applying $L \rightarrow R$ to t_n.

For a substitution rule, we first construct the part of t_{n+1} that does not include the subtree rooted at r_R. For an introduction rule, we take any path from the forest's root to r_L. Next, we construct S_L through r_L instead of S_R through r_R. This is possible since according to Corollary 1 all embedded graphs are trees, therefore the nodes of S_L are not already in t_n. We then look at the set of e-edges $(s,t) \in t_{n+1}$ such that $s \in S_R$ and $t \notin S_R$. Let (s,z) be such an edge originating from a d-edge d and S_z be the subtree rooted at z in t_{n+1}. Notice that S_z was already part of $\mathcal{F}_n$. s is tied to $s' \in S_L$ and therefore s' is a source node of d. We can expand t_n to include the edge (s',z) and S_z if s' is not already used with the d-edge d in t_n. This is guaranteed because d is not part of S_L (only d-edges that are not part of S_L are shared). Finally, we complete the construction of t_n by arbitrarily expanding any unused outgoing d-edge of t_n's nodes, until we obtain a complete embedded tree.

We constructed an embedded tree t_n in $\mathcal{F}_n$. Therefore, according to the inductive assumption, t_n is a consequent in the formalism. t_n contains S_L and an instantiation of the dual leaf variables. Therefore, it is matched by L and the rule $L \rightarrow R$ can be applied. It is easy to verify that an application of the rule on t_n will yield t_{n+1}, as required. Thus, t_{n+1} is also a consequent in the formalism. ∎

For the sake of simplicity, the above proofs ignored the case where one or more leaf variables in L that match multiple target nodes in l appear in R as non-leaves. As described in Section 5.2, in this case the matched target nodes are inserted to S_R as alternatives (with proper sharing of their modifiers). Consequently, S_R becomes a compact forest containing multiple trees. Similarly, S_L is a compact forest, whose represented trees correspond to the possible choices of matching the leaf variables. The mapping between the nodes matched by the leaf variables in S_L and the nodes generated for them in S_R defines a one-to-one mapping between the trees in S_L and S_R.

The above proofs can be easily adapted to handle this case, as follows. First, the proof of Lemma 1 need not change. In Theorem 1, the proof that a rule application does not create cycles still holds if the underlying graph of S_R is a DAG rather than a tree. To prove that each embedded partial DAG T' is a tree, we observe that exactly one of the trees embedded in S_R is part of T'. Thus, we can consider only that tree in S_R and its corresponding tree in S_L, while ignoring the rest of S_R and S_L, and proceed with the original proof. Similarly, to prove completeness in Theorem 2, we refer to the tree represented in S_L, which is part of t_n, and the corresponding tree in S_R. To prove soundness, we consider each of the subtrees in S_R and their corresponding tree in S_L.

References

Bar-Haim, R. (2010). *Semantic Inference at the Lexical-Syntactic Level*. Ph.D. thesis, Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel.

Bar-Haim, R., Berant, J., & Dagan, I. (2009). A compact forest for scalable inference over entailment and paraphrase rules. In *Proceedings of EMNLP*.

Bar-Haim, R., Berant, J., Dagan, I., Greental, I., Mirkin, S., Shnarch, E., & Szpektor, I. (2008). Efficient semantic deduction and approximate matching over compact parse forests. In *Proceedings of the TAC 2008 Workshop*.

Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., & Szpektor, I. (2006). The Second PASCAL Recognising Textual Entailment Challenge. In *The Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Bar-Haim, R., Dagan, I., Greental, I., & Shnarch, E. (2007). Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*.

Bar-Haim, R., Szpektor, I., & Glickman, O. (2005). Definition and analysis of intermediate entailment levels. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

Barzilay, R., & Lee, L. (2003). Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL*.

Barzilay, R., & McKeown, K. R. (2001). Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*.

Bensley, J., & Hickl, A. (2008). Workshop: Application of LCC's GROUNDHOG system for RTE-4. In *Proceedings of the TAC 2008 Workshop*.

Bentivogli, L., Clark, P., Dagan, I., Dang, H. T., & Giampiccolo, D. (2010). The Sixth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the TAC 2010 Workshop*.

Bentivogli, L., Dagan, I., Dang, H. T., Giampiccolo, D., & Magnini, B. (2009). The Fifth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the TAC 2009 Workshop*.

Berant, J., Dagan, I., & Goldberger, J. (2011). Global learning of typed entailment rules. In *Proceedings of ACL*.

Bhagat, R., & Ravichandran, D. (2008). Large scale acquisition of paraphrases for learning surface patterns. In *Proceedings of ACL-08: HLT*.

Bos, J., & Markert, K. (2005). Recognising textual entailment with logical inference techniques. In *Proceedings of EMNLP*.

Bos, J., & Markert, K. (2006). When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of The Second PASCAL Recognising Textual Entailment Challenge*.

Chklovski, T., & Pantel, P. (2004). VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP*.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*.

Connor, M., & Roth, D. (2007). Context sensitive paraphrasing with a single unsupervised classifier. In *ECML*.

Cooper, R., Crouch, R., van Eijck, J., Fox, C., van Genabith, J., Jaspars, J., Kamp, H., Pinkal, M., Milward, D., Poesio, M., Pulman, S., Briscoe, T., Maier, H., & Konrad, K. (1996). Using the framework. Tech. rep., FraCaS: A Framework for Computational Semantics.

Dagan, I., & Glickman, O. (2004). Probabilistic textual entailment: Generic applied modeling of language variability. PASCAL workshop on Text Understanding and Mining.

Dagan, I., Glickman, O., Gliozzo, A., Marmorshtein, E., & Strapparava, C. (2006a). Direct word sense matching for lexical substitution. In *Proceedings of COLING-ACL*.

Dagan, I., Glickman, O., & Magnini, B. (2006b). The PASCAL Recognising Textual Entailment Challenge. In Quiñonero-Candela, J., Dagan, I., Magnini, B., & d'Alché Buc, F. (Eds.), *Machine Learning Challenges. Lecture Notes in Computer Science, Vol. 3944*, pp. 177–190. Springer.

Dagan, I., Roth, D., Sammons, M., & Zanzotto, F. M. (2013). *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

de Salvo Braz, R., Girju, R., Punyakanok, V., Roth, D., & Sammons, M. (2005). An inference model for semantic entailment in natural language.. In *Proceedings of AAAI*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, *41*(6), 391–407.

Dinu, G., & Lapata, M. (2010). Topic models for meaning similarity in context. In *Proceedings of Coling 2010: Posters*.

Emele, M. C., & Dorna, M. (1998). Ambiguity preserving machine translation using packed representations. In *Proceedings of COLING-ACL*.

Fellbaum, C. (Ed.). (1998). *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. MIT Press.

Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of IJCAI*.

Ganitkevitch, J., Van Durme, B., & Callison-Burch, C. (2013). PPDB: The paraphrase database. In *Proceedings of HLT-NAACL*.

Giampiccolo, D., Magnini, B., Dagan, I., & Dolan, B. (2007). The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Giampiccolo, D., Trang Dang, H., Magnini, B., Dagan, I., & Dolan, B. (2008). The Fourth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the TAC 2008 Workshop*.

Glickman, O., & Dagan, I. (2003). Identifying lexical paraphrases from a single corpus: a case study for verbs. In *Proceedings of RANLP*.

Glickman, O., Shnarch, E., & Dagan, I. (2006). Lexical reference: A semantic matching subtask. In *Proceedings of EMNLP*.

Haghighi, A. D., Ng, A. Y., & Manning, C. D. (2005). Robust textual inference via graph matching. In *Proceedings of EMNLP*.

Harmeling, S. (2009). Inferring textual entailment with a probabilistically sound calculus. *Natural Language Engineering*, *15*(4), 459–477.

Heilman, M., & Smith, N. A. (2010). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of HLT-NAACL*.

Hickl, A. (2008). Using discourse commitments to recognize textual entailment. In *Proceedings of COLING*.

Hickl, A., & Bensley, J. (2007). A discourse commitment-based framework for recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Hickl, A., Bensley, J., Williams, J., Roberts, K., Rink, B., & Shi, Y. (2006). Recognizing textual entailment with LCC's GROUNDHOG system. In *The Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Jurafsky, D., & Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition* (Second edition). Prentice Hall.

Kamp, H., & Reyle, U. (1993). *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht.

Kay, M. (1996). Chart generation. In *Proceedings of ACL*.

Kazama, J., & Torisawa, K. (2007). Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of EMNLP-CoNLL*.

Kipper, K. (2005). *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Kouylekov, M., & Magnini, B. (2005). Tree edit distance for textual entailment. In *Proceedings of RANLP*.

Lehmann, J., Bizer, C., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., & Hellmann, S. (2009). DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*.

Lin, D. (1998). Dependency-based evaluation of minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC*.

Lin, D., & Pantel, P. (2001). Discovery of inference rules for question answering. *Natural Language Engineering*, *7*(4), 343–360.

Lotan, A., Stern, A., & Dagan, I. (2013). TruthTeller: Annotating predicate truth. In *Proceedings of HLT-NAACL*.

MacCartney, B., Galley, M., & Manning, C. D. (2008). A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*.

MacCartney, B., Grenager, T., de Marneffe, M.-C., Cer, D., & Manning, C. D. (2006). Learning to recognize features of valid textual entailments. In *Proceedings of HLT-NAACL*.

MacCartney, B., & Manning, C. D. (2009). An extended model of natural logic. In *Proceedings of IWCS-8*.

Macleod, C., Grishman, R., Meyers, A., Barrett, L., & Reeves, R. (1998). Nomlex: A lexicon of nominalizations. In *Proceedings of Euralex98*.

Maxwell III, J. T., & Kaplan, R. M. (1991). A method for disjunctive constraint satisfaction. In Tomita, M. (Ed.), *Current Issues in Parsing Technology*. Kluwer Academic Publishers.

Mehdad, Y., & Magnini, B. (2009a). A word overlap baseline for the recognizing textual entailment task. Unpublished manuscript.

Mehdad, Y., & Magnini, B. (2009b). Optimizing textual entailment recognition using particle swarm optimization. In *Proceedings of the 2009 Workshop on Applied Textual Inference*.

Melamud, O., Berant, J., Dagan, I., Goldberger, J., & Szpektor, I. (2013). A two level model for context sensitive inference rules. In *Proceedings of ACL*.

Meyers, A., Reeves, R., Macleod, C., Szekeley, R., Zielinska, V., & Young, B. (2004). The cross-breeding of dictionaries. In *Proceedings of LREC*.

Mi, H., Huang, L., & Liu, Q. (2008). Forest-based translation. In *Proceedings of ACL-08: HLT*.

Mirkin, S., Dagan, I., & Pado, S. (2010). Assessing the role of discourse references in entailment inference. In *Proceedings of ACL*.

Mirkin, S., Dagan, I., & Shnarch, E. (2009). Evaluating the inferential utility of lexical-semantic resources. In *Proceedings of EACL*.

Moldovan, D. I., & Rus, V. (2001). Logic form transformation of WordNet and its applicability to question answering. In *Proceedings of ACL*.

Nairn, R., Condoravdi, C., & Karttunen, L. (2006). Computing relative polarity for textual inference. In *Proceedings of International workshop on Inference in Computational Semantics (ICoS-5)*.

Pang, B., Knight, K., & Marcu, D. (2003). Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL*.

Pantel, P., Bhagat, R., Coppola, B., Chklovski, T., & Hovy, E. (2007). ISP: Learning inferential selectional preferences. In *Proceedings of HLT-NAACL*.

Ponzetto, S. P., & Strube, M. (2007). Deriving a large-scale taxonomy from wikipedia. In *Proceedings of AAAI*.

Ravichandran, D., & Hovy, E. (2002). Learning surface text patterns for a question answering system. In *Proceedings of ACL*.

Ritter, A., Mausam, & Etzioni, O. (2010). A latent dirichlet allocation method for selectional preferences. In *Proceedings of ACL*.

Romano, L., Kouylekov, M., Szpektor, I., Dagan, I., & Lavelli, A. (2006). Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL*.

Ron, T. (2006). Generating entailment rules based on online lexical resources. Master's thesis, Computer Science Department, Bar-Ilan University.

Ruppenhofer, J., Sporleder, C., Morante, R., Baker, C., & Palmer, M. (2009). Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*.

Saint-Dizier, P., & Mehta-Melkar, R. (Eds.). (2011). *Proceedings of the Joint Workshop FAM-LbR/KRAQ'11. Learning by Reading and its Applications in Intelligent Question-Answering*.

Schoenmackers, S., Etzioni, O., Weld, D. S., & Davis, J. (2010). Learning first-order horn clauses from web text. In *Proceedings of EMNLP*.

Shinyama, Y., Sekine, S., Sudo, K., & Grishman, R. (2002). Automatic paraphrase acquisition from news articles. In *Proceedings of HLT*.

Shnarch, E., Barak, L., & Dagan, I. (2009). Extracting lexical reference rules from Wikipedia. In *Proceedings of ACL-IJCNLP*.

Snow, R., Jurafsky, D., & Ng, A. Y. (2006a). Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING-ACL*.

Snow, R., Vanderwende, L., & Menezes, A. (2006b). Effectively using syntax for recognizing false entailment. In *Proceedings of HLT-NAACL*.

Stern, A., & Dagan, I. (2011). A confidence model for syntactically-motivated entailment proofs. In *Proceedings of RANLP*.

Stern, A., & Dagan, I. (2014). Recognizing implied predicate-argument relationships in textual inference. In *Proceedings of ACL*.

Stern, A., Stern, R., Dagan, I., & Felner, A. (2012). Efficient search for transformation-based inference. In *Proceedings of ACL*.

Szpektor, I., & Dagan, I. (2007). Learning canonical forms of entailment rules. In *Proceedings of RANLP*.

Szpektor, I., & Dagan, I. (2008). Learning entailment rules for unary templates. In *Proceedings of COLING*.

Szpektor, I., & Dagan, I. (2009). Augmenting WordNet-based inference with argument mapping. In *Proceedings of ACL-IJCNLP Workshop on Applied Textual Inference (TextInfer)*.

Szpektor, I., Dagan, I., Bar-Haim, R., & Goldberger, J. (2008). Contextual preferences. In *Proceedings of ACL-08: HLT*.

Szpektor, I., Tanev, H., Dagan, I., & Coppola, B. (2004). Scaling web based acquisition of entailment patterns. In *Proceedings of EMNLP*.

Tatu, M., Iles, B., Slavick, J., Novischi, A., & Moldovan, D. (2006). COGEX at the Second Recognizing Textual Entailment Challenge. In *The Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Tatu, M., & Moldovan, D. (2006). A logic-based semantic approach to recognizing textual entailment. In *Proceedings of COLING-ACL*.

Tatu, M., & Moldovan, D. (2007). COGEX at RTE3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Valencia, V. S. (1991). *Studies on Natural Logic and Categorial Grammar*. Ph.D. thesis, University of Amsterdam.

van Deemter, K., & Kibble, R. (2000). On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics*, *26*(4), 629–637.

Voorhees, E. M., & Harman, D. (1997). Overview of the sixth Text REtrieval Conference (TREC-6). In *Proceedings of TREC*.

Wang, M., & Manning, C. (2010). Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of COLING*.

Wang, R., & Neumann, G. (2007). Recognizing textual entailment using a subsequence kernel method. In *Proceedings of AAAI*.

Yates, A., & Etzioni, O. (2009). Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research (JAIR)*, *34*, 255–296.

Zanzotto, F. m., Pennacchiotti, M., & Moschitti, A. (2009). A machine learning approach to textual entailment recognition. *Natural Language Engineering*, *15*(4), 551–582.