A Compact Forest for Scalable Inference over Entailment and Paraphrase Rules

Roy Bar-Haim[§], **Jonathan Berant**[⋆], **Ido Dagan**[§]

[§]Computer Science Department, Bar-Ilan University, Ramat Gan 52900, Israel
{barhair,dagan}@cs.biu.ac.il

[⋆]The Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel
jonatha6@post.tau.ac.il

Abstract

A large body of recent research has been investigating the acquisition and application of applied inference knowledge. Such knowledge may be typically captured as *entailment rules*, applied over syntactic representations. Efficient inference with such knowledge then becomes a fundamental problem. Starting out from a formalism for entailment-rule application we present a novel packed data-structure and a corresponding algorithm for its scalable implementation. We proved the validity of the new algorithm and established its efficiency analytically and empirically.

1 Introduction

Applied semantic inference is concerned with deriving target meanings from texts. In the textual entailment framework, this is reduced to inferring a textual statement (the *hypothesis* h) from a source *text* (t). Traditional formal semantics approaches perform such inferences over logical forms derived from the text. By contrast, most practical NLP applications operate over shallower representations such as parse trees, possibly supplemented with limited semantic information about named entities, semantic roles etc.

Most commonly, inference over such representations is made by applying some kind of transformations or substitutions to the tree or graph representing the text. Such transformations may be generally viewed as *entailment (inference) rules*, which capture semantic knowledge about paraphrases, lexical relations such as synonyms and hyponyms, syntactic variations etc. Such knowledge is either composed manually, e.g. WordNet (Fellbaum, 1998), or learned automatically.

A large body of work has been dedicated to learning paraphrases and entailment rules, e.g. (Lin and Pantel, 2001; Shinyama et al., 2002; Szpektor et al., 2004; Bhagat and Ravichandran, 2008), identifying appropriate contexts for their application (Pantel et al., 2007) and utilizing them for inference (de Salvo Braz et al., 2005; Bar-Haim et al., 2007). Although current available rule bases are still quite noisy and incomplete, the progress made in recent years suggests that they may become increasingly valuable for text understanding applications. Overall, applied knowledge-based inference is a prominent line of research gaining much interest, with recent examples including the series of workshops on *Knowledge and Reasoning for Answering Questions (KRAQ)*[1] and the planned evaluation of knowledge resources in the forthcoming 5^{th} Recognizing Textual Entailment challenge (RTE-5)[2].

While many applied systems utilize semantic knowledge via such inference rules, their use is typically limited, application-specific, and quite heuristic. Formalizing these practices seems important for applied semantic inference research, analogously to the role of well-formalized models in parsing and machine translation. Bar-Haim et al. (2007) made a step in this direction by introducing a generic formalism for semantic inference over parse trees. Their formalism uses entailment rules as a unifying representation for various types of inference knowledge, allowing unified inference as well. In this formalism, rule application has a clear, intuitive interpretation as generating a new sentence parse (a *consequent*), semantically entailed by the source sentence. The inferred consequent may be subject to further rule applications

[1] http://www.irit.fr/recherches/ILPL/kraq09.html
[2] http://www.nist.gov/tac/2009/RTE/

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1056–1065,
Singapore, 6-7 August 2009.

and so on. In their implementation, each consequent was generated explicitly as a separate tree.

Following this line of work, our long-term research goal is to investigate effective utilization of entailment rules for inference. While the formalism of Bar-Haim et al. provides a principled framework for modeling such inferences, its implementation using explicit generation of consequents 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 entailment 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 as each rule can either be applied or not, independently. Indeed, we found that this exponential explosion leads to poor scalability in practice. Intuitively, we would like that each rule application would add just the entailed part (e.g. *kids*) to a packed sentence representation. Yet, we still want the resulting structure to represent a set of entailed sentences, rather than some mixture of sentence fragments whose semantics is unclear.

As discussed in section 5, previous work proposed only partial solutions to this problem. In this paper we present 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 (section 3). Our work is inspired by previous work on packed representations in other fields, such as parsing, generation and machine translation (section 5). As we follow a well-defined formalism, we could prove that all inference operations of Bar-Haim et al. are equivalently applied over the compact forest. We compare inference cost over compact forests to explicit consequent generation both theoretically (section 3.4), illustrating an exponential-to-linear complexity ratio, and empirically (section 4), showing improvement by orders of magnitude. These results suggest that our data-structure and algorithm are both valid and scalable, opening up the possibility to investigate large-scale entailment rule application within a well-formalized framework.

2 Inference Framework

This section briefly presents a (simplified) description of the tree transformations inference formalism of Bar-Haim et al. (2007). Given a source text, syntactically parsed, and a set of *entailment rules* representing tree transformations, the formalism defines the set of consequents derivable from the text using the rules. Each consequent is obtained through a sequence of rule applications, each generates an intermediate parse tree, similar to a proof process in logic.

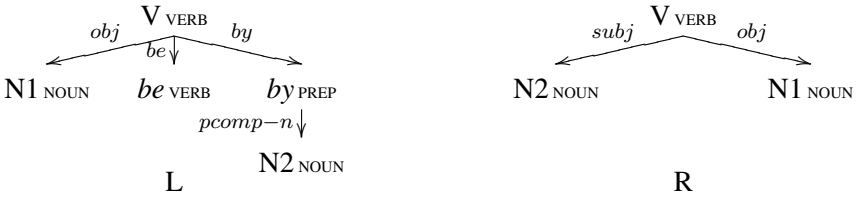
More specifically, sentences are represented as dependency trees, where nodes are annotated with lemma and part-of-speech, and edges are annotated with dependency relation. A rule '$L \rightarrow R$' is primarily composed of two *templates*, termed *left-hand-side* (L), and *right-hand-side* (R). Templates are dependency subtrees which may contain POS-tagged *variables*, matching any lemma. Figure 1(a) shows passive-to-active transformation rule, and (b) illustrates its application.

A rule application generates a derived tree d from a source tree s through the following steps:

L matching: First, a match of L in the source tree s is sought. In our example, the variable V is matched in the verb *see*, $N1$ is matched in *Mary* and $N2$ is matched in *John*.

R instantiation: Next, 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 defining alignments explicitly, 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 *beautiful* (modifying *Mary*) is copied for $N1$.

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



(a) Passive to active transformation (substitution rule)

Source: Beautiful Mary was seen by John yesterday.

Derived: John saw beautiful Mary yesterday.

(b) Application of passive to active transformation

Figure 1: An inference rule application. POS and relation labels are based on Minipar (Lin, 1998)

of l's nodes) with r. This is the case for the passive rule, as well as for lexical rules such as '*buy* → *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 r.

In addition to inference rules, the formalism includes *annotation rules* which add features to existing parse tree nodes. These rules have been used for identifying contexts that affect the polarity of predicates.

As shown by Bar-Haim et al., this concise, well defined formalism allows unified representation of diverse types of knowledge which are commonly used for applied semantic inference.

3 Efficient Inference over Compact Parse Forests

As shown in the introduction, explicit generation of consequents (henceforth *explicit inference*) leads to an exponential explosion of the number of generated trees. In this section we present our efficient implementation for this formalism. Our implementation is based on a novel data structure, termed *compact forest* (Section 3.1), which compactly represents a large set of trees. Each rule application generates explicitly only the nodes of the rule right-hand-side while the rest of the consequent tree is shared with the source, which also reduces the number of redundant rule applications. As we shall see, this novel representation is based primarily on *disjunction edges*, an extension of dependency edges that specify a set of alternative edges of multiple trees. Section 3.2 presents an efficient algorithm for inference over compact forests, followed by a discussion of its correctness and complexity (sections 3.3 and 3.4).

3.1 The Compact Forest Data Structure

A compact forest $\mathcal{F}$ represents a set of dependency trees. Figure 2 shows an example of a compact forest, containing both the source and derived sentences of Figure 1. 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; $rel_d : S_d \rightarrow \mathcal{D}$ is a function specifying the dependency relation corresponding 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 bottom-most d-edge in Figure 3. Then $S_d = \{$*of, like*$\}$, $T_d = \{$*candy, sweet*$\}$, $rel(of) =$ *pcomp-n*, and $rel($*like*$) =$ *obj*.

A d-edge represents, for each $s_i \in S_d$, a set of

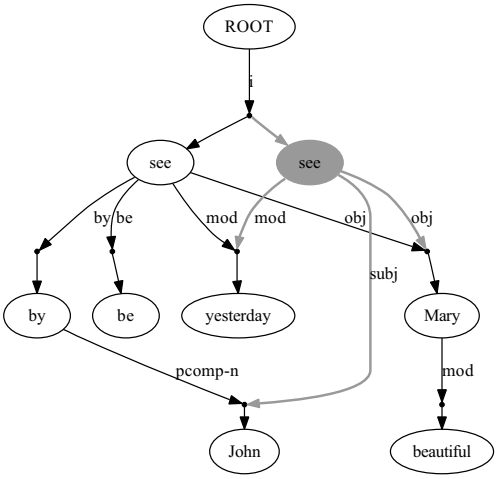


Figure 2: A compact forest containing both the source and derived sentences of Figure 1. Parts of speech are omitted.

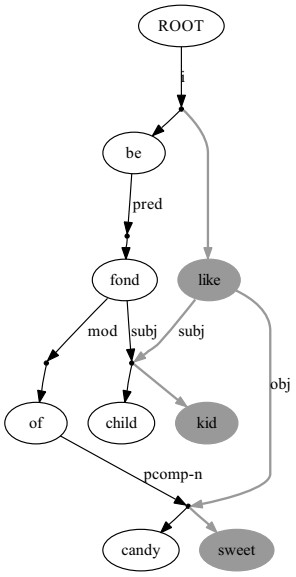


Figure 3: A compact forest representing the 2^3 sentences derivable from the sentence *"children are fond of candies"* using the following three rules: *'children→kids'*, *'candies→sweets'*, and *'X is fond of Y→X likes Y'*.

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* (notice that the definition implies that all source nodes in S_d have the same set of alternative target nodes T_d). 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 2, 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*, comprise the set of trees represented by $\mathcal{F}$.

Figure 3 shows another example for a compact forest efficiently representing the 2^3 sentences resulting from three independently-applied rules.

3.2 The Inference Process

Next, we describe the algorithm implementing the inference process of Section 2 over the compact forest (henceforth, *compact inference*), illustrating it through Figures 1 and 2.

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 $\mathcal{F}$. The black part of Figure 2 corresponds to the initial forest.

Rule application comprises the following steps:

L matching: 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 2. We denote by l the subtree of t in which L was matched. As in section 2, the match in our example is $(V, N1, N2)$=*(see, Mary, John)*. Notice that this definition does not allow l to be scattered over multiple embedded trees.

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 of substitutable types.

In this paper we further assume that all target nodes of the same d-edge have the same part-of-speech[3]. 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/sweeets*}' (cf. Figure 3), 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).

Derived tree generation: A template r consisting of R while excluding variables that are leaves of both L and R (termed *dual leaf-variables*)[4] is generated and inserted into $\mathcal{F}$. In case of a substitution rule (as in our example), r is set as an alternative to l by adding r's root to T_d, where d is the incoming d-edge of l's root. In case of an introduction rule, it is set as an alternative to the other trees in the forest by adding r's root to the target node set of the forest root's outgoing d-edge. In our example, r is the gray node (still 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 r (i.e. a non-dual leaf) is instantiated according to its match in L (as in Section 2), e.g. V is instantiated with *see*. As specified above, if the variable is a leaf in L is not a dual leaf then it is matched in a set of nodes, and hence each of them should be instantiated in r. This is decomposed into a sequence of simpler operations: first, r is instantiated with a representative from the set, and then we apply (ad-hoc) lexical substitution rules for creating a new node for each other node in the set[5].

Alignment sharing: Modifiers of aligned nodes are shared (rather than copied) as follows. Given a node n_L in l aligned to a node n_R in 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 $rel_d(n_R) = rel_d(n_L)$. This is illustrated by the sharing of *yesterday* in Figure 2. We also copy annotation features from n_L to n_R.

We note at this point that the instantiation of variables that are not dual leaves (e.g. V in our example) 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 is performed analogously 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 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 *beautiful Mary* and *John* are shared this way for variables $N1$ and $N2$.

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.

3.3 Correctness

In this section we present two theorems, which prove that the inference algorithm is a valid implementation of the inference formalism of Section 2. Due to space limitations, the proofs themselves are omitted, and instead we outline their general scheme.

We first argue that performing any sequence of rule applications over the set of initial trees results in a compact forest:

Theorem 1: *The compact inference process generates a compact forest.*

Proof scheme: We prove by induction on the number of rule applications. Initialization generates a single-rooted cDAG, whose embedded DAGs are all trees, as required. We then prove that if applying a rule on 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, in contradiction with the inductive assumption. A crucial observation for this proof is that for any path from a node u to a node v that passes through r, where u and v are

[3]This is the case in our current implementation, which is based on the coarse tag-set of Minipar (Lin, 1998).

[4]With the following exceptions: 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.

[5]Notice that these nodes, in addition to the usual alignment with their source nodes in l, share the same daughters in r.

outside r, there is also an analogous path from u to v that passes through l instead, QED.

Next, we argue that the inference process over a compact forest is complete and sound, i.e., it generates 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 embedded in a compact forest derivable from T by the compact 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 asserts that t_n is embedded in some derivable compact forest $\mathcal{F}$. It is easy to verify that applying r_n to $\mathcal{F}$ 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 $\mathcal{F}_{n+1}$. $\mathcal{F}_{n+1}$ was derived from the compact forest $\mathcal{F}_n$, using the rule r_n. It can be shown that $\mathcal{F}_n$ represents a tree t_n, such that applying r_n on t_n will yield t_{n+1} according to the formalism. The inductive assumption asserts that t_n is a consequent in the inference formalism and therefore t_{n+1} is a consequent as well, QED.

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

3.4 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 3 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.

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

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 that 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 3.

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, this already gives polynomial time complexity. In practice, indexing the forest nodes, as well as the typical low connectivity of the forest, result in a very fast matching procedure, as illustrated in the empirical evaluation, described next.

4 Empirical Evaluation

This section reports empirical evaluation of the efficiency of compact inference, tested in the recognizing textual entailment setting using the RTE-3 and RTE-4 datasets (Giampiccolo et al., 2007; Giampiccolo et al., 2009). These datasets consist of *(text, hypothesis)* pairs, which need to be classified as *entailing/non entailing*. Our first experiment shows, using a small rule set, that compact inference outperforms explicit inference by orders of magnitude (Section 4.1). 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 for a text (Section 4.2).

4.1 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). We used a set of manually-composed entailment rules for inference over generic linguistic phenomena such as passive, conjunction, relative clause, apposition, possessives, and determiners, which contains a few dozens of rules. To make a fair comparison, we aimed to make the explicit inference implementation reasonably efficient, e.g. by preventing generation of the same tree by different permutations of the same rule applications. Both 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.

The results are summarized in Table 1. 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. 3 out of the 100 pairs reached that limit with explicit inference, while the maximal node count for compact inference was only 268. The number of rule applications is reduced thanks 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.

4.2 Application to an RTE System

Experimental setting The goal of the second experiment was to assess that compact inference scales well for broad entailment rule bases. In this experiment we used the Bar-Ilan RTE system (Bar-Haim et al., 2009). The system operates in two primary stages: ***Inference***, in which entailment rules are applied to the initial compact forest $\mathcal{F}$, aiming to bring it closer to the hypothesis $\mathcal{H}$, and ***Classification***, in which a set of features is extracted from the resulting $\mathcal{F}$ and from $\mathcal{H}$ and fed into an SVM classifier, which determines entailment.

The classification setting and its features are quite typical for the RTE literature. They include lexical and structural measures for the coverage of $\mathcal{H}$ by $\mathcal{F}$, where high coverage is assumed to correlate with entailment, as well as features aiming to detect inconsistencies between $\mathcal{F}$ and $\mathcal{H}$ such as incompatible arguments for the same predicate or incompatible verb polarity (see below). For a complete feature description, see (Bar-Haim et al., 2009).

Rule Bases In addition to the *generic rules* described in Section 4.1, the following large-scale sources for entailment rules were used: ***Wikipeda:*** 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*'. ***WordNet:*** We extracted from WordNet (Fellbaum, 1998) lexical rules based on synonyms, hypernyms and derivation relations. ***DIRT:*** The DIRT algorithm (Lin and Pantel, 2001) learns from a corpus entailment rules between binary predicates, e.g. *'X is fond of Y→X likes Y'*. We used the version described in (Szpektor and Dagan, 2007), which learns canonical rule forms. ***Argument-Mapped WordNet (AmWN):*** A resource for entailment rules between verbal and nominal predicates (Szpektor and Dagan, 2009), including their argument mapping, based on WordNet and NomLexplus (Meyers et al., 2004), verified statistically through intersection with the unary-DIRT algorithm (Szpektor and Dagan, 2008). In total, these rule bases represent millions of rules. ***Polarity Annotation Rules:*** We compiled a small set of annotation rules for marking the polarity of predicates as *negative* or *unknown* due to verbal negation, modal verbs, conditionals etc. (Bar-Haim et al., 2009).

Search In this work we focus on efficient representation of the search space, leaving for future work the complementary problem of devising effective search heuristics over our representation. In the current experiment 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. 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 $\mathcal{H}$.

The system was trained over the RTE-3 development set, and tested on both RTE-3 test set and RTE-4 (which includes only a test set).

Results Table 2 provides statistics on rule application using all rule bases, over the RTE-3 development set and the RTE-4 dataset[6]. Overall, the primary result is that the compact forest indeed accommodates well extensive rule application 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.

The accuracies obtained in this experiment and the overall contribution of rule-based inference are shown in Table 3. 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 validate that the setting of our experiment represents a state-of-the-art system.

Inference over the rule bases utilized in our experiment improved the 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. As for our current experiment, we may conclude that the goal of assessing the compact forest scalability in a state-of-the-art setting was achieved [7].

Finally, Tables 4 and 5 illustrate the usage and contribution of individual rule bases. Table 4 shows the distribution of rule applications over the various rule bases. Table 5 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.

[6]Running time is omitted 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.

[7]We note that common RTE research issues, such as improving accuracy, fall out of the scope of the current paper.

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

	Accuracy		
Test set	No inference	Inference	Δ
RTE3	64.6%	66.4%	1.8%
RTE4	57.5%	60.6%	*3.1%

Table 3: 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).

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 4: 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.

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

Table 5: Contribution of various rule bases. Results show accuracy loss on RTE-4, obtained for removing each rule base (ablation tests).

5 Related Work

This section discusses related work on applying knowledge-based transformations within RTE systems, as well as on using packed representations in other NLP tasks.

RTE Systems Previous RTE systems usually restricted both the type of allowed transformations and the search space. Systems based on lexical (word-based or phrase-based) matching of h in t typically applied only lexical rules (without vari-

ables), where both sides of the rule are matched directly in t and h (Haghighi et al., 2005; MacCartney et al., 2008). The inference formalism we use is more expressive, allowing also syntactic and lexical-syntactic transformations as well as rule chaining.

Hickl (2008) derived from a given (t, h) pair a small set of *discourse commitments*, which are quite similar to the kind of consequents we derive by our syntactic and lexical-syntactic rules. The commitments were generated by several different tools and techniques, compared to our generic unified inference process, and commitment generation efficiency was not discussed.

Braz et al. (2005) presented a semantic inference framework which "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 provably equivalent to an expressive, well-defined formalism operating over individual trees, and each inferred consequent can be recovered from the compact forest.

Packed representations Packed representations in various NLP tasks share common principles, which also underly 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. In our work, alternatives are created 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 and Martin, 2008), and have been extended in later work for various purposes (Maxwell III and 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.

Packed representations have been utilized also 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), and thus their representation and algorithms are quite different from ours.

6 Conclusion

This work addresses the efficiency of entailment and paraphrase rule application. We presented a novel compact data structure and a rule application algorithm for it, which are provably a valid implementation of a generic inference formalism. We illustrated inference efficiency both analytically and empirically. Beyond entailment inference, we suggest that the compact forest would also be useful in generation tasks (e.g. paraphrasing). Our efficient representation of the consequent search space opens the way to future investigation of the benefit of larger-scale rule chaining, and to the development of efficient search strategies required to support such inferences.

Acknowledgments

This work was partially supported by the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886, the FBK-irst/Bar-Ilan University collaboration and the Israel Science Foundation grant 1112/08. The second author is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship. The authors would like to thank Yonatan Aumann, Marco Pennacchiotti and Marc Dymetman for their valuable feedback on this work.

References

Roy Bar-Haim, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*.

Roy Bar-Haim, Jonathan Berant, Ido Dagan, Iddo Greental, Shachar Mirkin, Eyal Shnarch, and Idan

Szpektor. 2009. Efficient semantic deduction and approximate matching over compact parse forests. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Rahul Bhagat and Deepak Ravichandran. 2008. Large scale acquisition of paraphrases for learning surface patterns. In *Proceedings of ACL-08: HLT*.

Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. 2005. An inference model for semantic entailment in natural language. In *Proceedings of AAAI*.

Martin C. Emele and Michael Dorna. 1998. Ambiguity preserving machine translation using packed representations. In *Proceedings of Coling-ACL*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. MIT Press.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Danilo Giampiccolo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2009. The Fourth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Aria D. Haghighi, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP*.

Andrew Hickl. 2008. Using discourse commitments to recognize textual entailment. In *Proceedings of COLING*.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition.

Martin Kay. 1996. Chart generation. In *Proceedings of ACL*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

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

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*.

John T. Maxwell III and Ronald M. Kaplan. 1991. A method for disjunctive constraint satisfaction. In Masaru Tomita, editor, *Current Issues in Parsing Technology*. Kluwer Academic Publishers.

A. Meyers, R. Reeves, Catherine Macleod, Rachel Szekeley, Veronkia Zielinska, and Brian Young. 2004. The cross-breeding of dictionaries. In *Proceedings of LREC*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Proceedings of NAACL-HLT*.

Yusuke Shinyama, Satoshi Sekine, Kiyoshi Sudo, and Ralph Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of Human Language Technology Conference (HLT 2002)*, San Diego, USA.

Eyal Shnarch, Libby Barak, and Ido Dagan. 2009. Extracting lexical reference rules from Wikipedia. In *Proceedings of ACL-IJCNLP*.

Idan Szpektor and Ido Dagan. 2007. Learning canonical forms of entailment rules. In *Proceedings of RANLP*.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of COLING*.

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

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventure Coppola. 2004. Scaling web based acquisition of entailment patterns. In *Proceedings of EMNLP*.