A Syntax-based Rule-base for Textual Entailment
and a Semantic Truth Value Annotator

MA thesis submitted by Amnon Lotan

Department of Linguistics, Lester and Sally Entin Faculty of the Humanities, Tel Aviv University, Israel

Prepared under the guidance of Prof. Ido Dagan of the Department of computer science, Bar Ilan University

and under the guidance of Dr. Roni Katzir of the Department of Linguistics, Tel Aviv University

March 7, 2013

Contents

1 Introduction 4

2 Previous Work 7

2.1 The use of Syntactic Knowledge in Entailment Systems . . 7
2.2 Utilized sources of Syntax-based Inference Rules . . . . . . . 8

3 Formalism of The Generic Syntax-Based Rule-Base 9

3.1 Sentence Representation . . . . . . . . . . . . . . . . . . . . . 9
3.2 Entailment Rules . . . . . . . . . . . . . . . . . . . . . . . . . 10
3.3 Overview of Entailment Rule Definition and Application . . 12

4 A Generic Entailment Rule-base 13

4.1 Rule Editing Tool and Compact Schema . . . . . . . . . . . . 14
4.2 Complementizer Insertion/Deletion . . . . . . . . . . . . . . . 16
4.3 Coordinations . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.4 IS-A Implications . . . . . . . . . . . . . . . . . . . . . . . . . 19
4.5 Possessive Constructions . . . . . . . . . . . . . . . . . . . . . 21
4.6 Sentential Constructions . . . . . . . . . . . . . . . . . . . . . 22
4.7 Extracting Relative Clauses . . . . . . . . . . . . . . . . . . . . 26
4.8 Substituting Determiners . . . . . . . . . . . . . . . . . . . . . 29
4.9 Case Correction . . . . . . . . . . . . . . . . . . . . . . . . . . 30
Abstract

Textual entailment (TE) is the semantic inference task which takes two text fragments, and determines whether one entails the other. It captures the semantic inferences needed by many text understanding applications. Practical TE applications usually adopt relatively low-level lexical or lexical-syntactic representations of text, which correspond closely to language structure. In many cases, such approaches miss out on some of the valuable and more abstract information, at the generic-syntactic and semantic levels.

This thesis first presents a novel comprehensive generic syntax-based rule-base for the TE task, which offers a wide variety of over 60 generic entailment rules that can substitute between copious equivalent or entailing constructions in categories such as: active vs. passive, coordination, apposition, determiners, possessives, and case correction. Also, they can extract simplified IS-A and HAS-A implications from over a dozen generic patterns, and successfully decouples relative clauses.

Based on inputs from previous salient works on the topic, as well as novel contributions, the rule-base is represented according to the Stanford Dependencies standard, is based on a well defined formalism, and is made publicly available for use with TE systems, including full documentation, and tools for rule design and software compilation. The rules are honed for high resilience to syntactic structure diversity, especially in RTE texts, which are used in the field’s most common benchmark.

Qualitative and quantitative depth evaluations are reported, including a manual dataset analysis, that demonstrate the high potential for knowledge resources of this type in TE, the wide coverage this rule-base has over the required set of syntax-based transformations in this setting, its high accuracy, and how it significantly improves the performance of a concrete entailment engine. Additionally, I show that much work is needed for such systems to make better use of syntax-based resources.

I then introduce TRUTHTELLER, a novel algorithm and system that takes the syntactic parse tree of a given sentence, and identifies the semantic truth value of each predicate and clause. In contrast with previous work, it is the first such system meant to serve the natural language inference research community as an open source tool, enriching conventional syntax tree representation. Some possible uses for these annotations are: inferring parts of a sentence from the whole, improving similarity (and contradiction) measures between texts, and improving the accuracy of entailment rule matching. As a side product, TRUTHTELLER also annotates negation, and a classification of predicate implication types (factive, implicative, etc.). The download package also includes a lexicon of predicates and their implication types, the largest of its kind and the first to be made publicly available. Both the lexicon and TRUTHTELLER’s annotations are shown to have good accuracy, recall and precision.
1 Introduction

Textual entailment (TE) is the semantic inference task which takes two text fragments, termed text \((T)\) and hypothesis \((H)\), and determines whether \(T\) entails \(H\), in the sense that a human reader would judge that \(H\) is most likely true, given \(T\) (Dagan et al. 2009). TE can be used to solve the core challenges of many semantic inference applications such as Question Answering (QA), Information Extraction (IE), Relation Extraction, Summarization and educational applications (Harabagiu and Hiki, 2006; Ravichandran and Hovy, 2002; Shinyama and Sekine, 2006; Romano et al. 2006; Harabagiu et al. 2007; Nielsen and Ward, 2007). In recent years it has been established as an important research area within Natural Language Processing (NLP). Notably, the annual PASCAL Recognizing Textual Entailment (RTE) challenges (Bentivogli et al. 2009, 2010b) serve as the main benchmark for evaluating TE applications.

As a general application-independent framework for natural language inference, TE has guided my work in the creation of a knowledge resource, specifying syntax-based semantic inferences. In addition, I have created an independent system for generating generic inferences based on semantic annotations, and for calculating truth values of clauses and predicates, discussed separately from the rest of this study, in Section 7.

Generally speaking, TE systems try to detect (or rule out) entailment, by assessing how the elements of the hypothesis can be “covered” by the information that is expressed in the text or can be implied from it. While simple systems attempt only at lexical matching between \(T\) and \(H\) (Shnarch et al. 2012; Clark and Harrison, 2010; MacKinlay and Baldwin, 2009), it has been shown that systems that consider also the syntactic structures in \(T\) and \(H\) yield improved performance (Cabrio et al. 2008; Wang and Neumann, 2008). Though these systems benefit from syntactic analysis, they need to cope with the challenge of a variety of distinct syntactic structures that express or imply the same meaning.

Some previous works have taken various measures to address this challenge. For instance, Bar-Haim et al. (2007) and de Salvo Braz et al. (2005) built banks of syntax-based entailment rules, that can convert text fragments with certain predefined structures into equivalent or entailed structures. For example, they encode the equivalence \(\text{Man bites dog} \iff \text{Dog is bitten by man}\). Using these rules, an entailment system has the means to straightforwardly compare \(H\) against many different syntactic forms of \(T\). I consider this to be a sound approach, but these rule-bases suffer from several practical drawbacks: they are limited in scope, not publicly available, and their development lacked substantial performance evaluation.

In this study, I extend these and other previous works to present a novel comprehensive knowledge resource of syntax-based inference rules. It offers the broadest scope of rules to date, covering the content of the previous work
as well as many novel rules, motivated by empirical analysis that is based on
the target RTE dataset. The resource is publicly-available\footnote{http://u.cs.biu.ac.il/~nlp/downloads/index.htm}, including documenta-
tion and an edit utility, which allows for maintaining and altering the exist-
ing rules, as well as developing extensions. Furthermore, its potential
and empirical contribution to the TE task is methodically evaluated, via
ablation tests that isolate its impact on system performance, as well as via
quantitative and qualitative error analyses.

To represent my rules, I use a generic parse-tree transformation formal-
ism, similar to that of Bar-Haim et al (2007), which extends common
tree-transformation rules, such as in DIRT \cite{Lin:2001}. The
formalism defines entailment rules, as well as how a rule can be \textit{applied}
to a text and generate new consequents. Thus, my rule-base can be used by
different types of inference architectures. I tested it in BiUTEE \cite{Stern:2011}.

My knowledge resource encompasses a wide variety of syntactic phe-
nomena, relevant to the conversion between semantically-equivalent forms,
as well as to the extraction of generic implications. These include: substitut-
ing one sentential construction with an equivalent (like active with passive
voice), possessive constructions, complementizer insertion and deletion, co-
ordination manipulation, extracting IS-A relations (like from appositions),
d determiner substitutions and case correction.

The rules are designed based on the Stanford dependency relations stan-
dard \cite{deMarneffe2008}, as implemented by the EasyFirst parser \cite{Goldberg2010}. Thus, they should be compatible as-
is with any inference system that uses that parser. Based on my experience,
I estimate that to utilize them in other systems that adopt the Stanford
representation but use a different parser, rather minimal adaptations
would be necessary, in order to tweak the rules to the chosen parser’s flavour of
parse errors. For all other systems, a modest amount of work should be
dedicated, in order to modify the rules according to a new representation
scheme and parser.

Some particular earlier works are sources for rules, as follows. Amoia
and Gardent \cite{Amoia:2008} developed a suite of 36 inference patterns revolving
around English adjectives, out of which 9 are generic and were adapted into
my formalism. Hearst \cite{Hearst92} and later Pantel et al \cite{Pantel:2004} presented 15
syntactic patterns that detect hyponyms (IS-A relations) in general domain
natural English texts, most of which were similarly incorporated in my rule-
base. Finally, Bar-Haim et al \cite{Bar-Haim:2007} attempted to develop a comprehensive
generic rule-base, with rules dealing with 7 different syntactic categories. As
opposed to the other sources for rules, they also presented a formalism that
defines entailment rules and their application, which I adopt in this thesis.

Besides existing sources, novel rules account for about half of my in-
ventory, and as mentioned above, their design is mostly based on an investigation of the generic operations that would be most useful in the RTE challenge datasets.

As opposed to all previous work in this field, I report a series of evaluations and analyses to assess the resource. They can be divided into three types: a) estimating the general potential of syntax-based resources to help in resolving RTE datasets; b) measuring the rule-base’s recall, assessing how many of the required syntax-based operations are supplied by it, and its precision, assessing how often a rule application yields a correct entailment; c) analysing its impact on an actual RTE system, in particular the above-mentioned system BiuTee, via error analysis and ablation tests.

In an effort to make my knowledge resource useful, we’ve made it publicly available. The distribution package offers all the rules reported and evaluated below, as well as a separate collection of several more rules that are in themselves precise (see below), but which did not perform well in the ablation tests, and therefore were set aside. I make a point of retaining the latter, since my empirical experience led me to conclude that distinct entailment tasks, and distinct genres of text, tend to benefit from different sets of entailment rules.

The rules are given in XML format, and come with a graphical editor that allows for an intuitive tree-based display, for ease of maintenance and modification, and for defining further rules. A new component in BiuTee is capable of loading the rules onto Java objects that may be used in its open source environment, in compliance with my formalism. In order to offer more portability, BiuTee also provides a command line utility that converts rules from my XML format to CoNLL format. Developed as part of this work, the software is accompanied by a user guide, giving full documentation of all aspects of the rule-base and its usage.

This work is structured as follows. Section 2 discusses the aforementioned previous work. Section 3 lays out the formalism of the resource: sentence representation and the formal definition for entailment rules, and for the way an inference system can apply a rule to text, in order to generate new entailed consequents. Section 4 presents the rules themselves, illustrating examples for each of its main categories. Section 5 reports the extensive analyses and evaluations mentioned above, and Section 6 offers some conclusions.
2 Previous Work

2.1 The use of Syntactic Knowledge in Entailment Systems

Many research approaches represent and analyse $T$ and $H$ at the lexical level, with no syntactic analysis. Nevertheless, they all lack the ability to make inferences based on structure. For instance, *The horse was eating by the hay* and *The horse was eaten by the hay* would seem equivalent lexically, but their syntactic structures are distinct.

Many other approaches rely on some kind of syntactic representation of text (see below), and so they have the advantageous ability to identify equivalent, or contradicting, meanings that are expressed in the syntax, rather than at the lexical level. However, these systems face the challenge of sentences that are different in syntactic form, but express related meanings. E.g., *Mica wants this car* and *This is the car Mica wants* both mean that Mica wants this car.

One way, in which syntax-sensitive systems rise to this challenge, is to predefine a collection of transformations\(^3\) that allow conversion from one syntactic form to another, while preserving, or entailing, the meaning. Such transformations can be used to convert $T$ into a form more similar to $H$, or vice versa.

Below, I mention several entailment systems that incorporate syntax-based transformations, usually termed syntax-based entailment rules, in their entailment recognition.

\[\text{de Salvo Braz et al (2005)}\] incorporated syntax- and semantic-based entailment rules in a comprehensive entailment system. In their system, entailment rules are applied over hybrid syntacto-semantic structures called concept graphs. When the template side (left hand side, LHS) of a rule is matched in the concept graph (e.g., a pattern matching a passive voice construction), the graph is augmented with an instantiation of the right hand side (RHS) of the rule (e.g., an active voice construction). After several iterations of rule applications, the system attempts to embed the hypothesis in the augmented graph.

\[\text{Bar-Haim et al (2007)}\] presented another comprehensive entailment system that applies all matching rules on a syntactic parse tree of $T$, and iteratively on all consequents, thus representing all possible entailments within a novel data structure, called a compact forest of consequent trees. Compact forests are a scalable data structure, capable of representing a large set of distinct parse trees. If the forest contains trees that are both close enough to $H$ and contain no mismatches, as judged according to a special entailment classification process, then the system classifies it as a positive entailment. Otherwise, it is a non-entailment.

\(^3\)I use the term transformations in a broader sense than is normally used in much of the theoretical linguistics literature.
derived from a given $T - H$ pair a small set of consequents that he terms *discourse commitments*. These consequents are based on syntax ( coordinations, appositions, relative clauses etc), co-reference, predicate-complement structure, the extraction of certain relations, and several paraphrases acquired from the Web. The commitments were generated by several different tools and techniques. Pairs of commitments derived from $T$ and $H$ were fed into the next stages of the RTE system - lexical alignment and entailment classification.

All the above works share several disadvantages: the syntax-based rules were relatively few, were not made publicly available, nor were they evaluated against an objective benchmark.

In comparison, my research presents a comprehensive syntax-based rule-base, by far the largest of its kind. It contains extensive novel work, principally motivated by an RTE dataset analysis, while covering the contents of each of the previous works (see below). Furthermore, this study reports a series of methodical qualitative and quantitative evaluations, that measure the rule-bases potential, quality, and empirical usefulness for a particular entailment system.

2.2 Utilized sources of Syntax-based Inference Rules

As noted, many of the rules presented in this study are novel, based on an analysis of the required syntax-based rules in an RTE dataset, as well as on well known syntactic equivalences and entailments. Other rules were adopted from several papers from the Semantics and Natural Language Inference literatures, as follows.

Amoia and Gardent (2008) produced an adjective-oriented test suite of 36 adjectival inference patterns, based on substantial literature about classification of adjectives, on WordNet relations (Fellbaum, 1998; Miller, 1995) and on their own semantic classification of English adjectives. Their goal was to conduct an in-depth study of this narrow slice of inference problems, and to create a resource supporting the evaluation of computational systems handling natural language inference. In my rule-base, presented in Section 4, only 9 of their inference patterns are included, since the rest are not purely syntactic, i.e., they depend on lexical information, and thus fall out side the scope of the phenomena handled by my resource. For instance, I adopt the following transformations:

This is *Adj* for a $N \Rightarrow$ This is an *Adj* $N$

- *This is expensive for a carpet* ⇒ *This is an expensive carpet*

$N_1$ is *Adj* as a $N_2 \Rightarrow N_1$ is a *Adj* $N_2$

- *John is good as a cook* ⇒ *John is a good cook*
Inspired by their work, a novel set of similar rules in the adverbial domain was composed. Hearst (1992) outlined a way to discover hyponyms (IS-A relations) in general domain natural English texts, using generic lexical-syntactic patterns. A few such patterns were incorporated in the rule-base here, for instance:

\[ I \text{ like tomato juice and other delicacies} \Rightarrow \text{Tomato juice is a delicacy} \]

Pantel et al. (2004) present and evaluate several similar patterns, also used here.

While syntax-based transformations have been addressed in these and other works to some extent, so far only Bar-Haim et al. (2007) and Bar-Haim (2010) have attempted to develop a comprehensive generic rule-base. My syntax-based rule-base was initially modelled after theirs, which was available to me. However, as mentioned above, my resource possesses over twice as many rules and addresses more syntactic phenomena. Beyond this quantitative contribution, it is the first to be made publicly available, with tools for editing and augmenting it. As an additional methodological contribution, I present a manual dataset analysis, conducted to assess the potential of syntax-based resources in general, in-depth evaluations of the quality of my resource, and error analyses inspecting how a state of the art entailment system utilizes it.

3 Formalism of The Generic Syntax-Based Rule-Base

This section lays out how texts are represented in my system, the formalism defining the rule-base, and how it may be used in an entailment system.

3.1 Sentence Representation

This subsection describes the standards, tools and conventions my rule-base uses to represent texts.

My approach assumes that $T$ and $H$ are represented by dependency parse trees, detailed as follows. Two example dependency trees are shown in Figure 1(b). Nodes represent words and hold a set of features, including the word lemma and part-of-speech (POS). Edges are annotated with syntactic dependency relations (subject, auxiliary, etc.). From here on, I will use $T$ and $H$ to denote either the text and hypothesis, or their respective parse tree representations.

The syntactic parser I use to generate parse trees from plain text is EasyFirst (Goldberg and Elhadad, 2010), which is state of the art. It

---

4 EasyFirst scored 91.4 on unlabelled, (and 88.33 on labelled), dependency accuracy for
complies with the commonly used standards for POS tags of the Penn Tree Bank [Marcus et al. 1993] and the Stanford dependency relations standard for dependency parse representation (de Marneffe and Manning, 2008).

Each of my rules is defined using two parse tree templates (see below), represented using Stanford dependencies, while adopting a reduced version of the Penn Tree Bank POS tag set, comprised of: NOUN, VERB, ADJECTIVE, ADVERB, PREPOSITION, DETERMINER, PRONOUN, PUNCTUATION, and OTHER. The conversion from Penn POS tag set to the reduced POS tag set is straightforward, e.g., NN(simple noun) → NOUN, NNP(proper noun) → NOUN, VBN(participle verb) → VERB etc. The full conversion table is in Appendix B.

In addition, I neglect surface word forms in favour of their lemmas. This helps me assume a further degree of abstraction in the tree representation, in which I do not deal with variation in tense, aspect, mood, number and person. On the other hand, this means my rules cannot make inferences that are sensitive to any of the above morphological features. For instance, in my framework, the following text fragments all give identical entailments: Dana plays, Dana play, Dana has been playing, Dana playing, Dana to play.

Nonetheless, in the future, more linguistic data may be added to the structure of rule nodes and/or edges, thus expanding the formalism of my entailment rules (defined below), and enabling the rules to address more fine-grained linguistic phenomena.

3.2 Entailment Rules

This subsection presents the function of entailment rules, while the next subsection gives a more detailed definition.

Linguistic knowledge required for inference is represented in my framework as entailment rules, which encode parse tree transformations. Each rule application on the source text tree $T$ generates a new consequent sentence (represented as another parse tree). As a framework, I adopt Bar-Haim et al. (2007)'s definition, illustrated in Figure 1(a), with a sample entailment rule, representing a passive-to-active transformation. Nonetheless, for convenience and brevity, many entailment rules are presented in this study in textual format, rather than with full trees.

From a knowledge representation and usage perspective, entailment rules provide a simple unifying formalism for representing and applying a very
Figure 1: Application of an inference rule. POS and relation labels are based on the Stanford dependency standard. N1, N2 and V are variables, which are implicitly aligned between L and R, as described in Bar-Haim et al. (2009). Notice that the word *yesterday* is syntactically ambiguous in the sentence in Figure 1(b), since it could be interpreted as modifying either the embedded clausal predicate *seen*, or the sentence main predicate *rained*. In such ambiguous cases, I always rely on the parser’s interpretation as the de facto parse.
<table>
<thead>
<tr>
<th>Rule Type</th>
<th>Sources</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax-based</td>
<td>Typically manually-composed</td>
<td>Active/passive, apposition, relative clause, coordination</td>
</tr>
<tr>
<td>Lexical-Syntactic</td>
<td>Learned with unsupervised algorithms, and derived automatically by integrating information from lexical resources like WordNet and Nomlex</td>
<td>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</td>
</tr>
<tr>
<td>Lexical &amp; Statistical</td>
<td>Extracted from large manually constructed lexical resources like WordNet, Wikipedia</td>
<td>Steal ⇒ take, Albanian ⇒ Albania, Janis Joplin ⇒ singer, Amazon ⇒ South America</td>
</tr>
</tbody>
</table>

Table 1: Representing diverse knowledge types as entailment rules

broad range of inference knowledge. Some examples of this breadth are illustrated in Table 1.

Given the syntactically parsed tree $T$ of the source text, and a set of entailment rules, this formalism defines how to compute the set of consequents (entailed trees) 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. According to the formalism, a text $T$ is judged as entailing a hypothesis $H$, if $H$ is a consequent of $T$.

### 3.3 Overview of Entailment Rule Definition and Application

This subsection gives an overview of the formalism for applying entailment rules (described above) on parse trees, which I assume entailment systems follow. For the full technical details, see Appendix A. As an illustration, Figure 1(a) shows passive-to-active transformation rule, and Figure 1(b) shows its application.

A rule ‘$L \Rightarrow R$’ is composed of two templates, $L$ on the left-hand-side (LHS) and $R$ on the right-hand-side (RHS). Templates are dependency parse subtrees which may contain variable nodes alongside regular parse tree nodes. These variables are regular nodes that have no specified lemma, so that they match any lemma. A pair of corresponding variables on both sides of a rule (bearing the same variable name) are said to be aligned, meaning that, when the rule is applied, the contents and subtree of the LHS variable is copied to the RHS. For instance, in Figure 1(a) the two $N1$ variables are aligned, as are two pair of $N2$s and $V$s.
A rule may be applied to the tree of a parsed text, in which case matches between the entire LHS pattern and parts of the tree are sought. A valid match is essentially some segment of the tree, which corresponds to the entire LHS, both structurally and by the content of each corresponding node requiring equivalence in POS, relation to parent and (for non-variables) lemma. If no match is found, the rule application terminates. Otherwise, each match triggers an RHS instantiation, where data in the tree is copied to the RHS template via the rule’s alignments, if any, so that the RHS contains no variables, and a new consequent tree is generated, as follows.

At this point, there are two different ways for the consequent generation to complete, depending on whether the rule type is substitution, or extraction. If substitution, the instantiated RHS is embedded within a copy of the original tree, replacing the subtree that was matched against the LHS, like in the example in Figure 1(b). If extraction, the instantiated RHS is left as is, constituting a new tree, whose root is the root of the instantiation. For example, applying the rule in Figure 2 to the lower tree of Figure 1(b) yields the proposition John saw beautiful Mary yesterday.

In Section 4 when presenting my rule-base, it is stated at the beginning of each subsection, which of the two types is used.

4 A Generic Entailment Rule-base

This section presents the rules of my resource, categorized by syntactic phenomena. For considerations of scope and fluency, each category is described in summary, with only a sample of its rules detailed and discussed. The full resource is available for download along with its full specification, further documentation, and software for formulating new rules and for compiling rules into Java and CoNLL representations.

In order to acquire independent empirical motivation for new rules, I performed a preliminary dataset analysis, as follows. First, I manually derived proofs for several dozen T - H pairs from a textual entailment oriented dataset, RTE5 main task (Bentivogli et al, 2009), and annotated in detail all inference phenomena involved in those proofs. I then observed the usage
counts of syntactic operations, and motivated several new rules by them. This informal experiment’s method is the same as in the final data analysis reported in Subsection 5.1. I mention this empirical motivation with regards to specific rules in this section, where relevant.

This section is built as follows: Subsection 4.1 describes my graphical rule editing tool, and the set of conventions with which I use it to represent many similar rules in one compact diagram, known as rule schemas; Subsection 4.2 presents my complementizer insertion and deletion rules; Subsection 4.3 presents my coordination construction rules; Subsection 4.4 presents my ISA relation extraction rules; Subsection 4.5 presents my possessive construction rules; Subsection 4.6 presents my general sentential constructions rules; Subsection 4.7 presents my relative clause rules; Subsection 4.8 presents my determiners related rules; and Subsection 4.9 presents my last rules, which correct errors in pronoun case marking left by the application of other rules.

4.1 Rule Editing Tool and Compact Schema

Since my representation of text and of rules is fine grained, while naturally the underlying inference pattern behind each rule is more general, it is common to have to define up to several dozen formal rules in order to capture all the various forms in which the pattern may occur. For instance, the operation of swapping places between a pair of conjuncts (e.g., Telma and Louise \(\iff\) Louise and Telma) is essentially the same whether the conjuncts are nouns, verbs, adjectives or adverbs, but it would take 4 different formal rules to cover all those variations. Hence, in order to allow capturing similar syntactic structures in one file, and, at once, to make a voluminous bank of manual rules more scalable and simple to understand, my graphical rule-editing tool and compilation component support a schematic representation of rules, called rule schema, that is more compact than my formal rules.

Figure 3 shows how I represent my active/passive conversion rule (discussed in Subsection 4.6), and demonstrates some of the main features of the rule schema. First, I use the figure to explain how some of the features of formal rules are represented.

At the top, the ruleType=substitution label means that this is a substitution rule, in contrast to an extraction rule. The two small nodes on the top labeled LHS and RHS are artificial nodes serving to point at the actual root nodes of the LHS and RHS, their two respective children. Notice how each LHS node must specify the relation (rel, except for the roots), POS tag (tag) and lemma (lemma, except for variable nodes) that it matches against. There are also three alignment relations (arrows labelled copy), stretching between pairs of corresponding nodes from LHS to RHS. An alignment relation indicates that the contents of the matched text node, along with its unmatched children (those not represented in the LHS), will be copied to the RHS counterpart, when generating the consequent tree.
Now I turn to explain some of the traits of my compact rule schema. Formally, each RHS node must also specify the same three properties as LHS nodes, which describe the text node it instantiates, except that, in the schema, the editor need not specify any single RHS node property, if that property is specified in the aligned LHS node. In these cases, at compilation step, the missing properties in the RHS node are supplemented using the properties of the aligned LHS node. For instance, the two bottom RHS nodes that have alignments, do not specify their tag. Furthermore, the bidirectional label means that this schema actually represents two rules, one for each direction of entailment, and the compiler will produce the two respective rules out of this one schema.

In addition, this rule exhibits multiple choice, in two of its nodes’ parameters, denoted by a ‘\’ between values, in order to compact several nearly identical rules into a single representation. More specifically, since I know it is possible for the subject (nsubj) to be either a noun or a pronoun, e.g., *Tim wrote a paper and later he published it*, it follows that both options must be listed in the tag parameter of the LHS’s subject node. The same goes for the direct object (dobj) and for their two alignments: the “by-subject” (with the relation pobj), and the passive-subject (nsubjpass).

A multiple choice parameter specifies two (or more) unambiguous rules, each containing only one of the listed options. It then follows that the LHS subject and object nodes have the combined effect of defining 4 concrete rules, one for each choice in the Cartesian product of their tag parameters.

Having said that, it is worth highlighting another aspect of multiple choice parameters. Notice the tag of each of the said RHS nodes is omitted, to signify that it gets its POS tag from its LHS mapped counterpart. Consider this example:

---

**Figure 3:** Graphical representation of the active-passive rule schema. See elaborations in the text.
(1) Some DNA evidence solved the case $\iff$ The case was solved by some DNA evidence

In this case, if the word evidence, matched in the LHS, is tagged (by the POS tagger) as a noun, it will remain a noun in the instantiated sentence (and if it were a pronoun, its RHS instantiation would be a pronoun).

In other cases, I may want an LHS node to match against not one or two POSs, but many, or even all, POSs, so I defined a wildcard value: $\text{tag}={}^{*}$

A similar wildcard was defined for relations. One additional feature of the rule schema is exemplified in Subsection 4.7.

Once a schema is completed manually, using the graphical editing tool, I run it through a compilation procedure that can produce several formal rules, and output them in CoNLL format. Accordingly, throughout this section rules are described in terms of compact schemas. For further documentation and code, the reader should refer to the download package.

4.2 Complementizer Insertion/Deletion

(Substitution) Complementizers are used in English to open subordinate clauses, and they constitute a closed lexical class that can be contained in a short list (that, who, when...). For many clausal constructions, which can usually be straightforwardly identified in the parse tree, the complementizer can either be present (materialized) or absent (implied) - without altering meaning. This motivates five rules in this category that insert and delete complementizers, such as the following:

(2) Insert or delete any relative pronoun + be at a position introducing a reduced object relative passive clause. Relative pronouns include that, which, who and whom.

(a) The horse which was raced past the barn fell

(b) The horse raced past the barn fell

Notice that the entailment The horse which was being raced past the barn fell is also entailed by (2a), since my formalism ignores grammatical aspect.

Figure 4 shows the graphical representation of (2). The LHS tree is designed to match against sentences like (2a) using four nodes. The match criteria of the root LHS node consists solely of requiring that the part of speech (POS) tag be a (any) noun. Its child node requires a “relative clause modifier” ($\text{rcmod}$) syntactic relation to its parent, and must be a verb. The two children of the $\text{rcmod}$ are not variables, but rather concrete words. The left child is either the word who or which, the only two possible relative pronouns for this structure. Accordingly, it takes a PRONOUN POS tag
Figure 4: The graphical representation of the reduced relative clause rule.

(which is the reduced category for 'WH pronouns', see Appendix B), and a \texttt{nsubjpass} relation, since it is the nominal subject of a passive clause. Its sibling represents any of the two common English passive auxiliary verbs “to be” or “to get”, with the \texttt{auxpass} relation.

On the other side, the RHS tree (which in this bidirectional schema also represents the LHS of the reversed rule) needs only two variable nodes to capture reduced relative clauses like in (2b). Its root node matches against any noun, and its child is a verb, which (according to the parser’s analysis) heads a participle modifier clause node (\texttt{partmod}).

(2) also exemplifies caveats in the rule. For example, it is prone to creating many ungrammatical sentences from \( T \), e.g.,

\[
* \text{The horse who was raced past the barn fell}
\]

Nonetheless, it is almost guaranteed that these would never be matched against a given \( H \), taken from a natural language corpus. Therefore, it seems beneficial to accept such caveats, for the sake of better coverage. Many other rules in my resource have caveats like this as well, which stem from limitations inherent to my particular formalism, or to any precise model describing rich variations in natural language.

Similar rules in this category were defined with other complementizers, such as \textit{that}, \textit{when}, \textit{whom}, in similar constructions, with or without the \textit{be} auxiliary.

4.3 Coordinations

\textit{(Substitution)} The coordination (conjunction) structure coordinates two or more syntactically equivalent phrases into a single phrase. For instance, the sentence

\[
\text{He likes her and she likes him}
\]
exhibits a coordination of two clausal phrases (CPs). In the Stanford dependency standard, conjunction words (and, or, but...) are all identifiable by their cc relation.

**Conjunct Deletion** Semantically and syntactically speaking, each conjunct is (usually) omissible, say in:

\[
\text{Two students and one teacher sing} \Rightarrow \text{Two students sing}
\]

Logically, this reasoning is wrong in many cases, for example:

\[
\text{Exactly two students and one teacher sing} \not\Rightarrow \text{Exactly two students sing}
\]

However, at present, it does not seem feasible to fine tune the criteria for conjunct deletion, since it would involve deep semantic and pragmatic information, such that cannot be reliably detected with the state of the art tools, and that would not be possible to contain within my formalism.

On the other hand, an inspection of occurrences of coordinations in TE datasets (in the preliminary analysis and in other sources) shows that problematic cases are scarce, and therefore conjunct deletion is expected to be predominately beneficial as is. My assumption of high empirical accuracy, for these and all other rules, is tested in Subsection 5.2.2.

Motivated by this observation and by similar work by Bar-Haim et al (2009), I specified several rules that delete any one conjunct in any distinguishable level of coordination - clausal, verbal, nominal, prepositional, adjectival and adverbial.

**Conjunct Swapping** Much like deleting a conjunct, in most cases swapping two conjuncts preserves entailment, for instance,

\[
(3a) \quad \text{Tom stayed home, but Jerry went out} \Leftrightarrow \text{Gerry went out, but Tom stayed home}
\]

but not

\[
(3b) \quad \text{Tom got lost, but Jerry found him} \not\Leftrightarrow \text{Gerry found him, but Tom got lost}
\]

I still assume that this operation is empirically beneficial, for motivations similar to the previous case, and developed three rules to address it. Each of these rules is tailored to a different variant of coordination.

Figure 5 shows the first of the three. It addresses the structure of (3a), which is the simplest. To understand it, it is important first to note the way the Stanford dependencies standard represents coordinations. This is reflected in the three nodes of the LHS tree, as follows:

a) A node representing the first conjunct, with two children, defined in
b) and c).
b) A child with the cc syntactic relation, stating that it is the conjunction function word.

c) A child with the conj relation. This is the second conjunct. If there are further conjuncts, each is represented by another such child of the first conjunct.

At first the LHS of the rule attempts to match against a coordination in the text, i.e., against the first conjunct and any one of the others. Afterwards, the rule swaps the positions of the two conjuncts in the RHS. If a coordination in a given text has three or more conjuncts, then matches of successive applications of the rule can yield all possible permutations of conjuncts.

4.4 IS-A Implications

(Extraction) This section presents extraction rules that extract IS-A relations out of $T$. Such canonical IS-A relations are straightforward to exploit in many settings. What’s more, in the RTE datasets in particular, many $H$s correspond to IS-A extractions.

Implied IS-A relations with Hearst Patterns The 10 rules in this group are adapted from Hearst (1992) and Pantel et al (2004). Each describes a composite noun phrase (NP) structure containing at least two NP components, extracts them, and uses them in a new (possibly reversed) copular sentence. The NP structures are:

1. NP1 such as NP2
2. Such NP1 as NP2

*I note that this might be an artefact of the choices made by the RTE dataset annotators, when manually picking the $T - H$ examples*
3. NP1 or other NP2
4. NP1 and other NP2
5. NP1 known as NP2
6. NP1 especially NP2
7. NP1 like NP2
8. NP1 including NP2
9. NP1-sg is (a OR an) NP2-sg
10. NP1-sg (a OR an) NP2-sg

For instance, applying the rule of pattern no. 1 will yield extractions like:

I like builders such as Bob ⇒ Bob is a builder

**Apposition**  Notice that the last pattern describes the generic apposition structure, labelled by the parser with the syntactic relation **appos**, and dealt with in this rule:

(4) Apposition to copula - extract an NP and its apposition to an independent IS-A construction, in **both** orders.

(a) *EU enlargement commissioner lent support to prominent Turkish novelist, Orhan Pamuk*

  ↓

(b) *Orhan Pamuk is prominent Turkish novelist*

  ↓

(c) *Prominent Turkish novelist is Orhan Pamuk*

This rule was also motivated by the the preliminary manual analysis.

In the above example, the indefinite determiner *a* appears to be missing from the two entailments. This is because this rule cannot account for the kind of determiner required by the noun phrase (*a, an, the*, etc.) within my generic formalism (which considers lemmas instead of surface forms), and therefore never adds one.

Figure 6 shows the apposition rule, for the case of the reversed order of the copular sentence, i.e., (4a) ⇒ (4c). The RHS in the figure describes a canonical copular sentence, like (4b) and (4c). Here, the LHS main NP becomes the subject, the LHS apposition becomes the main predicate, and it’s accompanied by a *to be* copular (**cop**) verb.

Other substitution rules in this category swap between an NP and its apposition, or delete an apposition, e.g.:
The ruleType = extraction label says this is an extraction rule. An apposition structure is identified straightforwardly, by a child node with the appos relation (in the LHS). Because there is no need to inspect the POSs, most nodes have a wildcard * tag.

- **EU enlargement commissioner lent support to prominent Turkish novelist, Orhan Pamuk** ⇔ **EU enlargement commissioner lent support to Orhan Pamuk, prominent Turkish novelist**

- **EU enlargement commissioner lent support to prominent Turkish novelist, Orhan Pamuk** ⇒ **EU enlargement commissioner lent support to Orhan Pamuk**

### 4.5 Possessive Constructions

(Substitution and extraction) There are two popular constructions in English to express possession. One uses an apostrophe-s marker, as in *a plane’s wings*, while another can be called an “of-construction”, as in *the wings of a plane*.

The two are usually semantically equivalent, so I developed a bidirectional substitution rule to convert one to another, shown in (5).

(5) Substitute an apostrophe-s construction with an “of-construction” and vice versa.

(a) Pamuk is Turkey’s best known writer ⇔ Pamuk is the best known writer of Turkey

In many cases, the same “of-construction” can be taken to have adjectival meaning, rather than possessive, as in

*The flood of a coal mine killed 10 in China* ⇔ *The coal mine flood killed 10 in China*

Another rule, with a compound rather than a possessive construction, performs this transformation.

In addition, I notice that a possessive construction immediately implies a HAS-A relation between its two components, similarly to the way an
apposition implies an IS-A relation. So I developed an extraction rule to extract independent HAS-A statements out of apostrophe-'s constructions, shown in (6) below. Notice how a similar HAS-A extraction from an “of-construction” is redundant, since an inference engine is expected to be able to perform (5) and then (6) sequentially.

(6) Extract both noun components of an apostrophe-'s’ construction to an independent HAS-A copula.

(a) Pamuk is Turkey’s best known writer ⇒ Turkey has a best known writer

4.6 Sentential Constructions

(Substitution) This subsection covers a relatively large number of bidirectional substitution rules, converting between diverse sentential constructions: active vs. passive, adjectival constructions, adverbial constructions and cleft constructions. These rules are aimed to convert some common constructions to and from the canonical subject-verb-object construction, where possible. This kind of “star topology” design, that maintains each construction at one step of entailment application away from its most canonical counterpart, minimizes and simplifies the chain of rule applications required to prove any complex inference.

Furthermore, most of the entailment rules described here have only theoretical motivation, and, to the best of my knowledge, have little or no empirical justification in the current RTE datasets. As an exception, the first rule does occur frequently, according to my preliminary data analysis.

Active-Passive Rule This rule converts between active and passive voice, e.g.:

(7) Some DNA evidence solved the case ⇐⇒ The case was solved by some DNA evidence

Notice there are many verbs for which the same transformation generates ungrammatical or unintelligible sentences, e.g.:

(8) My group escaped the police ⇐⇒ *The police was escaped by my group

Still, like in the discussion about the “delete conjunct” rule in Subsection 4.3, on the one hand, I cannot fine tune this rule further within the current formalism, and on the other hand, empirically it is shown to generate predominately valid entailment. The rule schema is shown in Figure 7.
The adjectival rules are an adaptation of work by Amoia and Gardent (2008), where they formulate dozens of syntax-based, lexical syntactic and semantic inference rules, all involving adjectives. I implemented and enhanced 9 of those cases that fit into my generic syntactic level framework. Like other rules in my study, these are not meant to be linguistically accurate, but rather are only expected to generate valid entailments on most texts.

Two of the rules are:

**(9) Predicative Attributive Construction**

This NP is Adj $\iff$ This is Adj NP

(a) (adjective) This table is red $\iff$ This is a red table

(b) (gerund) This chair is rocking $\iff$ This is a rocking chair

(c) (noun) This horse is a gift $\iff$ This is a gift horse

**Tough-Construction**

NP is Adj to V $\iff$ V NP is Adj $\iff$ It is Adj to V NP

(a) John is easy to please $\iff$ Pleasing John is easy $\iff$ It is easy to please John

Figure 8 gives a formal definition of a more generalized version of (9). It exhibits a few features common to many other rules in this subsection, meant to maximize the language diversity of matched texts, while obtaining a compact representation and minimizing false entailments.
First, note that the adjectival pattern in (9) does not seem to match against examples (9b) and (9c), which are verbal and nominal sentences (and note that, according to the Stanford dependencies representation, the head of a nominal/copular clause is the predicate, to the right of the copular verb). To accommodate for texts like this, I defined the POS tag of the root of the rule’s LHS node to be a choice between adjective, verb and noun, by setting \texttt{tag="ADJECTIVE\|VERB\|NOUN"}. Notice the root node of the RHS remains \texttt{NOUN} in all these examples.

Like in the active-passive rule, this node’s mapped RHS counterpart has no explicit \texttt{tag} parameter, so that the tag is copied through the alignment at compilation time.

Another general aspect demonstrated here is that loosening the match criteria of one node may require loosening other nodes as a result. In this case, comparing the parse trees of the above three examples, I noticed that the Stanford dependency relations standard labels the relation of the auxiliary verb \textit{to be} as copular (\textit{cop}) in adjectival and nominal sentences, while in verbal sentences it is regarded as an auxiliary (\textit{aux}). Hence the \texttt{rel="cop\|aux"} parameter in the auxiliary verb node in the LHS. The RHS’s auxiliary verb node only has the \texttt{rel="cop"} option, because the root of the nominal copular sentence is a noun.

For similar reasons, the RHS node mapped to the LHS root has multiple choice for the syntactic relation, between noun modifier and adjectival modifier, set as \texttt{rel="nn\|amod"}. The first option, when combined with the \texttt{NOUN} tag, matches parse trees of nominal examples like (9c). The second option, when combined with \texttt{ADJECTIVE} or \texttt{VERB}, matches parses of (9a) or (9b) type, respectively.

As the last point of interest in this rule schema, notice that the bottom node of the LHS has a multiple choice of words, covering the closed set of demonstrative pronouns, set by \texttt{lemma= "this\|that\|these\|those\|neither"}, in order to reflect the fact that any demonstrative can fit into this inference pattern.

In summary, this use of multiple parameter choice yields a compact rep-
representation of several similar rules into one manageable schema. Unfortunately, it also means that some anomalous rules will be compiled, for instance, one with a node that is both a nominal modifier by its syntactic relation and a verb by its POS tag. Luckily, in this case I can neglect the effect of such rules, because they can only be matched against specific ungrammatical texts, which are scarce in natural language datasets.

Adverbial Constructions  This subsection also covers a set of 4 novel bidirectional adverbial rules, inspired by adjectival rules, but which convert between sentential constructions involving adverbs. Some of these rules convert parts of speech, between corresponding adjectives and adverbs, for instance:

(11) Passive Adverbial

\[ \text{NP is Adv } V_{\text{passive}} \iff \text{NP is Adj}_{\text{adv}} \text{ to } V \]

\[(a) \text{ John is easily pleased } \iff \text{ John is easy to please}\]

In (11), \( \text{Adj}_{\text{adv}} \) denotes the adjective derived from the corresponding adverb. The seemingly lexical derivation between adverbs and adjectives is possible in a generic syntax-based knowledge resource, without the aid of a lexicon. This is done by exploiting a generalized assumption that, in most of these derivations, an adverb differs from its corresponding adjective solely in its POS tag and the \(-ly\) suffix of its lemma.

The other 3 rules are:

(12) In Construction

\[ \text{NP } V \text{ Adv } \iff \text{NP is Adj}_{\text{adv}} \text{ in Ving} \]

\[(a) \text{ John cooks splendidly } \iff \text{ John is splendid in cooking}\]

(13) Predicative Attributive Construction

\[ \text{NP}_{\text{subj}} V \text{ NP}_{\text{obj}} \text{ Adv } \iff \text{NP}_{\text{subj}} \text{ is Adj}_{\text{adv}} \text{ to } V \text{ NP}_{\text{obj}} \]

\[(a) \text{ John took this job foolishly } \iff \text{ John was foolish to take this job}\]

(14) Passive Construction with Subject

\[ \text{NP}_1 \text{ Adv } V \text{ NP}_2 \iff \text{NP}_2 \text{ is Adv } V_{\text{passive}} \text{ by } \text{NP}_1 \iff \text{NP}_2 \text{ is Adj}_{\text{adv}} \text{ for } \text{NP}_1 \text{ to } V \]

\[(a) \text{ Mary easily pleases John } \iff \text{ John is easily pleased by Mary } \iff \text{ John is easy for Mary to please}\]
Cleft Constructions  The last contribution to this subsection is based on some of the substantial literature available on cleft constructions. It contains 6 rules that substitute well known cleft constructions with the canonical subject-verb-object (SVO) construction, and vice versa, presented here:

(15) WH-Cleft
    (a)  What he wanted to buy was a Fiat ⇐⇒ He wanted to buy a Fiat

(16) Reversed WH-Cleft/Pseudo-Cleft
    (a)  A Fiat is what he wanted to buy ⇐⇒ He wanted to buy a Fiat

(17) It-Cleft
    (a)  It was Henry that kissed Rosie ⇐⇒ Henry kissed Rosie

(18) All-Cleft
    (a)  All he wanted to buy was a Fiat ⇐⇒ He wanted to buy a Fiat

(19) Inferential Cleft - Negative
    (a)  It is not that he loves her ⇐⇒ He doesn’t love her

(20) Inferential Cleft - Positive
    (a)  It’s just that he loves her ⇐⇒ He loves her

4.7 Extracting Relative Clauses
(Extraction) This subsection discusses a few rules that extract relative clauses to independent sentences.

It is well known that every definite noun phrase (NP) is presupposed, i.e., NPs modified by the, this, possessors etc. This signifies that the existence of every definite NP is entailed from the text, and, in case the NP has a relative clause, the clause’s body is entailed along with the modified NP as its subject or object. This kind of entailment persists even if the external argument is negated (not, never...), is questioned (perhaps, might...) or is conditioned (if X then...). For example:

(21) Isn’t the beer that you brought cold? ⇒ You brought beer

In contrary, most indefinite NPs are not presupposed, for example:

(22) A dollar saved is a dollar earned ⇒ A dollar was saved

7For example: http://en.wikipedia.org/wiki/Cleft_sentence
Ideally, I wish to form rules that extract relative clauses to independent sentences, like in (21), but not like (22). In practice, although I could form a few rules that detect definiteness of nouns, it is impossible to combine them all into one rule that also performs the clause extraction. Furthermore, under my formalism, I have neither the tools for annotating a feature like definiteness, nor can I form an entailment rule that matches against such an annotation.

So, instead of abandoning these rules altogether, while not breaching my simple formalism, I incorrectly assume in these rules that all relative clauses are presupposed. This trade-off is justified by extensive informal empirical tests on RTE datasets, and also by the formal rule-base evaluations, reported in Section 5.

According to the Stanford dependency standard, most relative clause constructions (like in (21)) can be characterized by three nodes:

1. The modified external noun
2. The main predicate of the clause, with an rcmod (relative clause modifier) relation to the external noun
3. (optional) The complementizer (which, who or that), with a subject or object relation to the clausal predicate, which is the same relation that should hold in the extracted entailment, from the noun to the predicate

In contrast, reduced participle relative clauses (like in (22)) are characterized by the partmod (participle modifier) relation from the clausal predicate to the modified noun.

The first rule, that covers many cases of relative clause constructions, where the modified noun is either the subject or the direct object (with no prepositions) of the external clause, like in (21), is illustrated in Figure 9. The figure warrants a few observations. In the LHS, the top node is the head noun that is modified by the relative clause. The second is the main predicate of the clause, using the idiom predicate_pos that stands for the multi choice of all possible predicate POS: “verb \ noun \ pronoun \ adjective”. The third is the complementizer of the clause, which, in this type of relative clause, can be either the subject, passive subject, or direct object of the predicate. Also, as mentioned above, its relation determines the RHS relation from the noun to the predicate, where the two are swapped. To implement this requirement, the relation is copied, at compilation time, over the copy_rel labeled arrow.

With this last feature I can summarize three different ways, in which the parameters of a schema RHS node are set at compilation stage, in decreasing order of precedence:
Figure 9: Graphical representation of the main relative clause extraction rule. In the second LHS node from the top PREDICATE_POS is an idiom for the multi choice of all possible predicate POS: “VERB\NOUN\PRONOUN\ADJECTIVE”. The copy_rel label in the bottom diamond is not an alignment (used at the RHS instantiation step), but rather it specifies that the rel parameter is copied at rule compilation stage, after the multi choice of the rel of the LHS node is expanded.

1. parameter values are copied through a copy_rel, copy_tag or copy_lemma arrow (not via alignment)

2. parameter values are written explicitly in the RHS node (as in most figures in this section)

3. parameter values are copied from an aligned LHS (with a COPY arrow)

Notice that this rule, like others in this category, does not cover many cases, in which the complementizer is not materialised, like in entailing (c) directly from (a) here:

a. I would like that book on the top shelf
   ⇓

b. I would like that book that is on the top shelf
   ⇓

c. That book is on the top shelf

This is deliberate, since complementizer insertion and deletion is covered independently by the rules in Subsection 4.2 Therefore, it is necessary to apply an intermediate complementizer insertion rule in (b).

Another rule addresses the reduced relative clause construction, e.g.

Roberta found the pack, hidden underneath the morning paper

Other rules cover texts with slight variations in the above two constructions, which cause the parser I use to consistently produce distinct erroneous trees.
These last rules are sensitive to the parser used, and not just to the Stanford standard.

Relative clauses of indirect complements, such as

*Roberta pushed aside the paper, under which the pack sat*

are varied and harder to characterize, so covering them is left for future work.

### 4.8 Substituting Determiners

(Substitution) This subsection features 8 rules that convert determiners, *a, the, some, not all, other, seven, Tom’s, etc.* Most entailments between determiner phrases are unidirectional, and most of the rules had at least some motivation in my preliminary data analysis. To my experience, the most widely applied rule from this group is:

(23) Substitute any of the determiners *the, that, this, other, another, these, those, the other and some* with *a*

(a) *Pluto is like other Kuiper Belt objects ⇒ Pluto is like a Kuiper Belt object*

At first, the next rule seems to generate bad sentences in practically every application:

(24) Substitute *an* with *a*, and vice versa

(a) *Pluto is like a Kuiper Belt objects ⇐⇒ Pluto is like an Kuiper Belt object*

However, since swapping between these two indefinite articles changes surface form but not meaning, this rule is useful in many common use cases that replace vowel-initial nouns with consonant-initial nouns. One use case is when employing my resource in tandem with a lexical-based rule-base (such as Wordnet), like in this case, where *honourable* is substituted with its synonym *respectable*:

*But Brutus is an honourable man ⇒ But Brutus is an respectable man*

Here my generic rule is beneficial to correct the error created by the lexical rule. Another common use case, relevant to many inference systems, is when *T* and *H* have different indefinite articles, and the system must assess the similarity between them, e.g.:

T: *But Brutus is an honourable man*

H: *But Brutus is a respectable man*

In this case, applying my rule may be is beneficial in making the two propositions more alike, without altering the meaning.
4.9 Case Correction

(Substitution) The need for the next two rules follows from many pronoun case errors created by other rules, e.g., the following error can be caused by an application of the active-to-passive rule:

They were escorted by the police ⇔ The police escorted they

This was also noted by Bar-Haim et al. (2007), who first formed these rules that detect incorrect pronoun case, and correct it, by substituting the appropriate lemmas. Since the closed set of pronouns is the only class of English nouns with visible case marking, the rules can be implemented generically, without the aid of a lexicon.

(25) Substitute an accusative pronoun in subject position with its nominative counterpart

(a) Her took the car ⇒ She took the car

(26) Substitute a nominative pronoun in object position with its accusative counterpart

(a) The car was taken by she ⇒ The car was taken by her

5 Evaluation and Analysis

I analysed and evaluated my approach and the new resource in three major aspects: 1) the potential impact of a resource of this type in the RTE datasets (Subsection 5.1); 2) the resource’s quality in terms of recall and precision (Subsection 5.2); and 3) its impact on a particular inference system (Subsection 5.3). To address these three aspects, I conducted a series of analytical and empirical experiments.

5.1 Manual Analysis of a Syntax-Based Resource Potential

My first analysis estimates the potential contribution of generic syntax-based inference rules to the TE task, i.e., how common are generic syntax-based inference phenomena in a manually computed “gold standard” of T – H proofs (in opposed to imperfect machine-aided proofs). In addition, I would like to give a quantitative prediction as to the relative usefulness of different kinds of rules.

I conducted my analysis over the most recent RTE dataset that was available to me at the time, RTE6 (Bentivogli et al. 2010b), as RTE is the main benchmark for TE technology. For the purposes of this analysis, it suffices to describe RTE as a collection of several thousand T - H pairs.
Most are not entailing (negative pairs), while about 5.6% of the pairs are positive.

I randomly sampled 50 positive $T - H$ pairs, and manually constructed their “gold standard” entailment proofs as a sequence of transformations that transforms $T$ to $H$, following the transformation-based inference paradigm of $\text{?}$, which follow Bar-Haim et al (2007) in its use of entailment rules. Each proof consists of a chain of entailment preserving atomic transformations, applied to the syntactic parse tree of $T$, ending in a tree identical to $H$’s tree (exact match). See example proof in Table 2.

The range of linguistic transformations I allowed myselfs to use can be broadly categorized according to their operational mechanism (substitution vs. extraction), and by the linguistic knowledge they use.

In terms of linguistic knowledge, a rule may rely either on syntactic knowledge (i.e., the rules described in this work), lexical or lexical-syntactic knowledge, on coreference resolution (replacing an anaphor with its antecedent) or bridging. Bridging is still a vague term in the literature, and I take it to mean transformations that introduce a subtree, copied from another tree within the context of $T$, which is implicitly referred to within $T$, see Mirkin et al, 2010.

An example for each rule category I employed is given in Table 3. For convenience, I demonstrate these categories with plain text and not with parse trees.

In cases where I could not come up with a plausible full proof for a pair, using the above transformation-based methodology, it was marked “hard”, set aside, and replaced by a new random positive pair, so that I end up with 50 full proofs. Overall, I assessed 7 pairs to be “hard”, and the following results refer to the remaining 50 fully annotated pairs.

5.1.1 Usage of Syntax-Based Rules as a Whole

I first investigate the overall need for syntax-based operations by counting the $T - H$ pair proofs that contain syntax-based transformations, and how many such operations each proof contains.

Table 4 presents the distribution of the pairs according to the number of syntax-based rule applications included in their proofs (57 applications in total). I see that the majority of the pairs (78%) did contain at least one syntax-based application. On average, each pair required 1.16 transformations. This analysis shows that generic syntax-based rules play a substantial part in the entailments of RTE datasets.

5.1.2 Profiling of Rules by Phenomena

In addition, I analysed the distribution of syntax-based rule applications over syntactic phenomena, to determine which rules are applied more frequently,
Jill Carroll was seized by gunmen

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule Category</th>
<th>Entailed T</th>
</tr>
</thead>
<tbody>
<tr>
<td>local coreference substitution</td>
<td>Coref</td>
<td>Carroll’s driver, quoted in a story posted on the Monitor’s website, said gunmen jumped in front of the car, pulled Jill Carroll from it, and drove off with their two captives all within 15 seconds</td>
</tr>
<tr>
<td>delete verbal conjunct</td>
<td>Syntactic Coordination</td>
<td>Carroll’s driver, quoted in a story posted on the Monitor’s website, said <strong>gunmen pulled Jill Carroll</strong> from it, and drove off with their two captives all within 15 seconds</td>
</tr>
<tr>
<td>Lexical Syntactic Substitution</td>
<td></td>
<td>Carroll’s driver, quoted in a story posted on the Monitor’s website, said <strong>gunmen seized Jill Carroll</strong>, and drove off with their two captives all within 15 seconds</td>
</tr>
<tr>
<td>active to passive</td>
<td>Syntactic Active-Passive</td>
<td>Carroll’s driver, quoted in a story posted on the Monitor’s website, said <strong>Jill Carroll was seized by gunmen</strong>, and drove off with their two captives all within 15 seconds</td>
</tr>
</tbody>
</table>
| say X → X             | Semantic Extraction    | Jill Carroll was seized by gunmen                                                                                                                                                               

Table 2: An example manually derived proof for a $T - H$ pair, from the RTE6 Dev main task dataset

32
<table>
<thead>
<tr>
<th>Category</th>
<th>Example Rule</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactic Substitution</td>
<td>Swap a Noun with Its Apposition</td>
<td>$T_1$ My friend, Ben, is coming over</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_2$ Ben, my friend, is coming over</td>
</tr>
<tr>
<td>Syntactic Extraction</td>
<td>Extract Apposition to Copula</td>
<td>$T_1$ My friend, Ben, is coming over</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_2$ Ben is my Friend</td>
</tr>
<tr>
<td>Lexical Substitutions</td>
<td>$Begin \Rightarrow Launch$</td>
<td>$T_1$ A peace process was <strong>begun</strong> by Pakistan and India</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_2$ A peace process was <strong>launched</strong> by Pakistan and India</td>
</tr>
<tr>
<td>Lexical Syntactic Substitution</td>
<td>$suffer from X \Rightarrow have a X$</td>
<td>$T_1$ Peter Jennings announced that he <strong>was suffering from</strong> lung cancer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_2$ Peter Jennings announced that he <strong>had a</strong> lung cancer</td>
</tr>
<tr>
<td>Lexical Syntactic Extractions</td>
<td>$In connection with X \Rightarrow There was X$</td>
<td>$T_1$ Bagri was arrested in <strong>connection with</strong> a blast at the airport</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_2$ There was a blast at the airport</td>
</tr>
<tr>
<td>Semantic Extraction</td>
<td>$Say that X \Rightarrow X$</td>
<td>$T_1$ The paper <strong>said</strong> the economy is recovering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_2$ The economy is recovering</td>
</tr>
<tr>
<td>Coreference Resolution and Bridging</td>
<td>Coreference Substitution</td>
<td>$T_1$ <strong>The two countries</strong> have been to war three times</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_2$ <strong>Pakistan and India</strong> have been to war three times</td>
</tr>
</tbody>
</table>

Table 3: Examples for inference rules and their application, one from each category used in the dataset annotation. $T_1$ is the text before application and $T_2$ is the entailed text.

<table>
<thead>
<tr>
<th># Syntax-based Operations in Pair</th>
<th># Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>11 (22%)</td>
</tr>
<tr>
<td>1</td>
<td>24 (48%)</td>
</tr>
<tr>
<td>2</td>
<td>12 (24%)</td>
</tr>
<tr>
<td>3</td>
<td>3 (6 %)</td>
</tr>
</tbody>
</table>

Table 4: The distribution of the solved $T - H$ pairs, according to the number of syntax-based transformations each proof contains.
Table 5: Distribution of generic rule applications, grouped by syntactic phenomena, as used in the manual analysis, and by the BiuTee entailment system (see discussion in Subsection 5.3). The figures are percentages of the total syntax-based applications.

and which are rare. To this end, I used the manual analysis to sum up the number of rules contained in the manually conducted proofs, grouped by syntactic phenomena.

The resulting distribution is given in the second column of Table 5 and it shows that some rule categories are used much more than others, while neither one is dominant. The third column is discussed in Subsection 5.3.

5.2 Resource quality

While the previous subsection evaluated the potential for my resource in the dataset, this subsection assesses its quality in terms of recall and precision.

By recall I mean the number of syntax-based rules that are covered by the resource, out of all the rules used in the manual analysis. I also tested the recall of the formalism, which is the number of syntax-based rules permitted in the formalism (defined in Section 3), out of all syntax-based rules used in the manual analysis. By precision I denote the probability that applying a syntax-based rule from the resource preserves entailment.

5.2.1 Recall

The recall measures are shown in Table 6. In total, I find that 21% of the listed applications require syntax-based rules which are not available in the rule-base, though all are allowed by the formalism. Therefore, the recall for the rule-base is estimated at 79%, while the recall for the formalism is estimated at 100%, for this sample.

Only 10 missing rules accounted for this gap, and I estimate that most of these are relatively rare and specific to this dataset, in comparison with the
Table 6: Recall for each investigated syntactic category, according to the manual dataset analysis.

<table>
<thead>
<tr>
<th>Syntactic Category</th>
<th># Apps in Annotation</th>
<th># Apps Covered by Formalism</th>
<th># Apps Covered by Rule-base (Recall)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apposition</td>
<td>15</td>
<td>15</td>
<td>13 (87%)</td>
</tr>
<tr>
<td>Relative Clause</td>
<td>11</td>
<td>11</td>
<td>7 (63%)</td>
</tr>
<tr>
<td>Determiners</td>
<td>11</td>
<td>11</td>
<td>7 (63%)</td>
</tr>
<tr>
<td>Possessive</td>
<td>8</td>
<td>8</td>
<td>8 (100%)</td>
</tr>
<tr>
<td>Active/Passive</td>
<td>7</td>
<td>7</td>
<td>7 (100%)</td>
</tr>
<tr>
<td>Coordination</td>
<td>3</td>
<td>3</td>
<td>3 (100%)</td>
</tr>
<tr>
<td>Adjectival</td>
<td>2</td>
<td>2</td>
<td>0 (0%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>57</strong></td>
<td><strong>57</strong></td>
<td><strong>45 (79%)</strong></td>
</tr>
</tbody>
</table>

The bulk of my existing rules, whose use is more widespread. Also, in Subsection 5.3.2 I show why many rules that have some, but not much, motivation, are likely to generate more false entailments than valid ones. For these reasons, I estimate that it would require a significant effort, and dozens of new rules, to raise the recall measure on most similar datasets.

5.2.2 Precision

I judge a given rule application to preserve entailment, if the underlying meaning of the tree generated by this application is entailed by the underlying meaning of the source tree, on which it was applied. To this aim, I randomly sampled 100 syntax-based rule applications from my resource, which were performed by a concrete RTE system, BiuTee (?), whose operation is described in Appendix C.

To produce the 100 samples of syntax-based rule applications for this analysis, I ran BiuTee on the RTE6 Test dataset, in one of its best configurations (?), along with my syntax-based resource. This configuration includes two knowledge resources, Wordnet (Fellbaum, 1998; Miller, 1995) and Directional Similarity (Kotlerman et al, 2010).

Out of the log of all rule applications, which were included in BiuTee’s proofs for the $T - H$ pairs in the test set, I randomly extracted 100 syntax-based rule applications, 50 from proofs of positive pairs, and 50 from negative

---

8I excluded “Case Correction” rules (Section 4.9), because I do not consider case alteration to change meaning, i.e., *he* and *him* are equivalent in this analysis.
Correct Applications  94%
Incorrect Applications
Rule Caveats  3%
Parser Errors  3%

Table 7: Precision (entailment preservation) analysis of a sample of 100 rule applications

pairs. These include both correct and incorrect proofs that the system generated, because I need to test for rule application precision in both contexts. I then judged whether each sampled rule application preserves entailment. Note that the more commonly used rules are more likely to be sampled and evaluated, and that this bias reflects how the resource is used in practice.

The results are shown in Table 7. Notice that the vast majority (94%) of rule applications are correct, i.e., their application preserves entailment. Of the few remaining errors, 3% are due to caveats in the rules themselves, mostly having to do with limitations of the formalism.

One such case involved the pair:

(27) T: Kashmir is divided between India and Pakistan, but both claim the region in its entirety

H: Kashmir is claimed in full by India and Pakistan

where T has a coordination construction at the matrix clause level. On this coordination, BiúTee applied a rule that matches correctly to it, but which is meant to be applied only on verbal coordinations, where the coordinated elements share a single subject. The rule first swaps between the two elements, and then reattaches the first element’s subject to the second one, e.g., Kids laugh and cry easily ⇔ Kids cry and laugh easily. Since it was applied to a clausal coordination, it yielded an ungrammatical and non-entailed sentence:

*Kashmir both claim the region in its entirety, but is divided between India and Pakistan

With this and several similar rules, I accept such caveats in the rule’s precision, in favour of higher coverage.

Furthermore, an additional 3% were errors caused by the parser, which produced ungrammatical or improbable parse trees, causing unexpected results when rules are applied over such erroneous parses.
5.3 Impact on BiuTee

In the previous two subsections I discovered a high potential for syntax-based rule-base resources in the RTE task, and assessed a high quality of my resource’s recall and precision. This subsection investigates to what degree does a state-of-the-art RTE system, namely BiuTee, take advantage of the resource’s potential.

The data used hereafter were obtained from two runs of BiuTee: a test run, configured the same as in Subsection 5.2.2, and a baseline run, without the use of my resource, for comparison.

5.3.1 BiuTee Usage

In order to enhance the rule profiling results in Subsection 5.1.2 with an empirical parallel, I collected usage counts of the resource’s rules from BiuTee’s application logs. These results, in the third column of Table 5, display a somewhat similar distribution to that of the manual analysis, in the second column.

5.3.2 BiuTee’s Utilization of the Rule-Base

Subsection 5.1 provided me with an estimation for the potential usefulness of a syntax-based rule resource on a given dataset, in terms of the rules’ usage rates. Now I compare this estimation with the way BiuTee actually applies rules from my resource, on the same dataset.

First I investigate the coverage, i.e., to what degree did the system reproduce the syntax-based applications used in the manual analysis, on the same $T \rightarrow H$ proofs. This experiment may help reveal system components which hamper better use of the rule-base, and problems with the rules themselves.

Second, I investigate the way BiuTee used the rules, as follows. Originally, each rule was designed to address a particular syntactic phenomenon. However, I observed that BiuTee sometimes applies a rule, in cases where a human would judge that the phenomenon takes no part in the needed inference, like in the following example:

(28) $H$: Ajaib Singh Bagri is a mill worker

$T$: ...Ripudaman Singh Malik, 57, and Kamloops, British Columbia mill worker Ajaib Singh Bagri, 55, are charged with multiple counts of conspiracy ... in the world’s worst airline terrorism act ...

Rule: Substitute determiners, the $\rightarrow$ a
<table>
<thead>
<tr>
<th>Applied / Why not applied</th>
<th>% of manually identified syntax-based rule applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied by the system</td>
<td>26.3%</td>
</tr>
<tr>
<td>Unavailable rule, within the formalism</td>
<td>29.8%</td>
</tr>
<tr>
<td>System found alternative proof</td>
<td>15.8%</td>
</tr>
<tr>
<td>Parser error</td>
<td>10.5%</td>
</tr>
<tr>
<td>System lacking prerequisite rules</td>
<td>7%</td>
</tr>
<tr>
<td>Resource lacking prerequisite rules</td>
<td>5.3%</td>
</tr>
<tr>
<td>Unavailable rule, out of the formalism</td>
<td>3.5%</td>
</tr>
</tbody>
</table>

Table 8: The distribution of cases in which BiuTee did not apply a syntax-based entailment rule, which was expected according to the manual annotation.

T’: ...Ripudaman Singh Malik, 57, and Kamloops, British Columbia mill worker Ajaib Singh Bagri, 55, are charged with multiple counts of conspiracy … in a world’s worst airline terrorism act …

In this case the system chose to apply a rule that substitutes determiners, and later to move the generated article a to another part of the sentence, to make T resemble H more, but it did so without any linguistic justification.

Coverage Referring back to the manual analysis data of Subsection 5.1 I checked, for each syntax-based rule application, whether it was also applied by BiuTee in the same proof. In those cases where BiuTee did not use the predicted rule application, I documented the reason for the transformation’s omission, based on the alternative proof that the system had found.

In the first line of Table 8 I see that in total, BiuTee reproduced only about 26% of the expected syntax-based rule applications (15 applications). The rest of the table presents a breakdown of the reasons for transformation omissions, which explain the low coverage in the first row. Examples illustrating each listed cause follow.

Unavailable rule, within the formalism - The largest contributors to the coverage gap are syntax-based transformations that were found necessary or beneficial in the manual analysis, and that are feasible according to my formalism, but were not included in the resource. The reason for this typically is either because they are inaccurate to the degree that I consider them to cause more false entailments than valid ones, or because I simply did not conceive of them.

As an example, consider:

T: ... a suitcase exploded at Japan’s Narita airport, killing two baggage handlers as they transferred it to Air India Flight 301 for
Bangkok and Delhi...

H: Two baggage handlers were killed while transferring suitcases to an Air India Flight.

Here the manual proof suggests applying a rule that converts the boldface segment of $T$ from active voice into passive voice, without the presence of a subject, making it similar to $H$, i.e., \textit{killing two handlers $\Rightarrow$ two handlers were killed}. This hypothetical rule would look like my existing active/passive rule (in the direction from active to passive), without the nodes representing its LHS and RHS subjects.

Nonetheless, a rule that matches such a loose participle construction does not exist in the rule-base, because it would be matched against many clauses that have an overt subject and cause false entailments like \textit{A blast killed two handlers $\Rightarrow$ *A blast two handlers were killed}. Note that the resource does have a similar rule for converting active to passive voice, but it requires the presence of a subject (Subsection 4.6). Example (27) in Subsection 5.2.2 offers a similar discussion about a rule-base rule.

System found alternative proof - BiuTee constructs alternative proofs, when it has the ability to repeat the entire proof from the manual analysis (i.e., the relevant entailment rules are available), but its proof search algorithm found another proof, without the syntax-based rule in question. Mostly, these alternative system proofs includes linguistically unmotivated transformations, but still gets an overall score that is just as good as the score of the proof I constructed.

For instance, consider the following simplified proof:

T: \textit{The blast occurred at a hotel in Taba}

H: \textit{A blast occurred at a hotel in Taba}

Naturally, at this point in the manual analysis, I chose to apply the first “substitute determiners” rule from Subsection 4.8 which replaced \textit{the} with \textit{a}. However, at the same point the system chose to perform a DUPLICATE AND MOVE operation, copying the second \textit{a} next to the position of the \textit{the}, yielding the same end result (full $T - H$ match), while the overall score of the proof was as good as my proof’s. Thus, the “substitute determiners” rule was not applied here, due to the alternative proof the system found.

Parser error - Parser errors produce many flawed syntactic trees, which prevent matching with a desired syntax-based rule. These account for 10.5% of such cases. For example, consider the pair:

T: \textit{All that changed Thursday, when two buses ... shuttled in each direction between Srinagar and Muzaffarabad, the capitals of the}
Indian and Pakistan-administered Kashmir...

H: Srinagar is the summer capital of Indian Kashmir.

The entailment proof I had composed in the manual analysis includes a rule application that converts the apposition construction in T (in bold) into an independent copular sentence, similar to H (the “Apposition to Copula” rule, Section 4.4). However, the parser used by BiUTee (EasyFirst) produced an erroneous analysis for this sentence, in which the apposition relation between Srinagar and Muzaffarabad and the capitals is missing. In this situation, the “Apposition to Copula” rule could not be matched against the text, although it is available in the rule-base and is needed for a correct proof.

System lacking prerequisite rules - To illustrate this category, consider the simplified pair:

T: Dick Cheney and Harry M. Whittington had been hunting ...

H: Harry M. Whittington is a hunter

In the manual analysis, I constructed a 2-step proof for this pair:

1. Apply the hypothetical lexical syntactic substitution rule (containing these explicit words) \(X \text{ hunts } \Rightarrow X \text{ is a hunter}\), generating Dick Cheney and Harry M. Whittington are (a) hunters

2. Apply the syntax-based rule “delete conjunct” (see Subsection 4.3), generating Harry M. Whittington is a hunter

Here, I neglect the effect of some auxiliaries and word inflections in T, as explained in Subsection 3.1. Since the knowledge resources loaded onto BiUTee didn’t have any the lexical syntactic rules similar to the one I used, apparently the system found no proof that featured the “delete conjunct” rule (with a high enough score). Therefore, this generic rule was not applied as expected.

Resource lacking prerequisite rules and Unavailable rule, out of the formalism - As a demonstration for the last two categories, consider this simplified pair:

T: India, Pakistan Due to Hold Meetings

H: Pakistan and India were due to hold meetings

T looks like the headline of a news item (short, capitalized words, no copula nor coordination words etc.), which hints me that the comma structure in boldface is a coordination. In this case, in my manual analysis I assumed
the existence of a generic rule that can substitute such comma constructions with coordinations, e.g., *India and Pakistan*. Appropriately, I proceeded to apply the conjectured rule, and then apply a “swap conjuncts” rule (see Subsection 4.3). These two applications yielded the desired construction *Pakistan and India*, and completed an almost full $T - H$ match.

Nonetheless, although the rule I conjectured is generic (does not depend on lexicon), it is impossible to form within my formalism, since it is suitable only for sentences that are headlines, and I have no systematic way to detect headlines. Therefore, that rule application was annotated “not applied - unavailable rule, out of the formalism”, while the application of the “swap conjuncts” rule was annotated “not applied - resource lacking prerequisite rules”.

**Justified vs. Spurious Rule Applications** I examined the cases in which BruTEE applied rules spuriously, i.e., where the rule’s syntactic phenomenon is not relevant to the correct inference chain. For example, in the above-mentioned pair:

$$T: \textit{All that changed Thursday, when two buses ... shuttled in each direction between Srinagar and Muzaffarabad, the capitals of the Indian and Pakistan-administered Kashmir}...$$

$$H: \textit{Srinagar is the summer capital of Indian Kashmir}$$

applying an “active to passive” rule on $T$ would be spurious, since a human would judge that converting between active and passive is irrelevant to this inference. Such cases occur when applying the rule produces some side-effect, which can be further exploited by the system to apply other rules that ultimately help construct an erroneous proof. I investigate how frequently such applications were performed.

In order to obtain a fuller account of such cases, I revisited the manual analysis of syntax-based rule applications from Section 5.2.2 focusing on the 46 entailment-preserving rule applications (according to the precision evaluation there), in proofs of positive pairs. For each rule-application I drew a new judgement: whether the application was spurious, and if so, why.

The results are presented in Table 9. First, I see that about half of the syntax-based rule applications were judged as justified and the other half were spurious. This may indicate that the main challenge for future work lies not in improving the rules themselves, but in developing better entailment system components that would apply them properly.
Table 9: The contribution of justified vs. spurious syntax-based rule applications

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Justified rule use</td>
<td>48%</td>
</tr>
<tr>
<td>Spurious use</td>
<td></td>
</tr>
<tr>
<td>System decision</td>
<td>50%</td>
</tr>
<tr>
<td>Parser Error</td>
<td>2%</td>
</tr>
</tbody>
</table>

5.3.3 Impact on system results

In this subsection I assess the impact of the resource on BiuTee’s overall performance, beginning with a breakdown of the results into categories of $T - H$ pairs, and ending with a discussion on the global performance measures.

Breakdown of Ablation Results  First I closely examine the resource’s impact on the ratio of correct vs. mistaken system classifications (positive/negative entailment). The data is taken from an ablation test, comparing BiuTee’s classifications with and without the syntax-based rule-base, configured the same as in Section 5.2.2.

I compared each system classification (positive/negative) with the gold standard annotation (true/false), thus classifying each answer into one of four categories: true-positive, false-positive, true-negative or false-negative.

Based on these statistics, for each $T - H$ pair I combined the labels from the two runs, yielding eight composite categories, shown in the two left columns of Table 10. The left label describes the result from the run without syntax-based rules, while the right part describes the run with them. Notice that other combinations are not possible, e.g., a TP cannot become a FP by altering the classification for the same pair. Presenting the answers this way can highlight the resource’s impact in finer detail.

The main results of the ablation runs are presented in the third column of Table 10, “Total System Answers”. The vast majority of answers fall in the top 4 rows, which means they are not affected by the resource. The bottom 4 rows show that syntax-based rules affected merely 2% of the system’s answers (91 out of 4353). Within those 91, use of the resource improved about two thirds of the answers (from FN to TP, and from FP to TN), while in a third of the cases it was the other way around (60 vs. 31).

Finally, in order to enhance these statistics, I counted in the fourth column the number of pairs whose proof (when the syntactic rule-base is

---

9 The examination involves only 4535 pairs out of the total 19972 pairs in the RTE6 Test dataset, since BiuTee judged the others to be non entailing based on an information retrieval ranking filter, which is used at a preprocessing stage.
### Table 10: Ablation of BiuTee’s classifications for RTE6 Test, with and without the syntactic rule-base. Each answer label is composed of the T( rue)/F(alse)/P(ositive)/N(egative) combination of the baseline run’s classification on the left, and the run with the syntactic resource on the right.

<table>
<thead>
<tr>
<th>Classification Label</th>
<th>w/o resource</th>
<th>with resource</th>
<th>Classifications for Proofs Involving Generic Transformations</th>
<th>Classifications for Proofs Not Involving Generic Transformations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>System Classifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TP</td>
<td>TP</td>
<td>382</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>FP</td>
<td>FP</td>
<td>273</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>FN</td>
<td>FN</td>
<td>227</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>TN</td>
<td>TN</td>
<td>3562</td>
<td>1814</td>
</tr>
<tr>
<td></td>
<td><strong>Subtotal</strong></td>
<td><strong>4444</strong></td>
<td><strong>2250</strong></td>
<td><strong>2194</strong></td>
</tr>
<tr>
<td></td>
<td>FN</td>
<td>TP</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>TN</td>
<td>FP</td>
<td>21</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>FP</td>
<td>TN</td>
<td>49</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>TP</td>
<td>FN</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td><strong>Subtotal</strong></td>
<td><strong>91</strong></td>
<td><strong>58</strong></td>
<td><strong>33</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td></td>
<td><strong>4535</strong></td>
<td><strong>2308</strong></td>
</tr>
</tbody>
</table>

The figures in the two right columns seem to suggest the latter conclusion.

**Global Ablation Results** Table 1 shows the global F1, recall and precision measures of the system classifications without the syntax-based resource (baseline configuration, with WordNet and Bap knowledge resources) and with it, on the RTE5 and RTE6 datasets, which were the two most recent RTE datasets available to me at the time of the evaluation. I use the standard definitions of precision, recall and F1 used in NLP benchmarks.

I see that straightforwardly using the resource on RTE6 slightly enhances
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Recall</th>
<th>Precision</th>
<th>F1</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline RTE6</td>
<td>43.07%</td>
<td>55.37%</td>
<td>48.45%</td>
<td>N/A</td>
</tr>
<tr>
<td>Rule-base Test RTE6</td>
<td>43.17%</td>
<td>57.71%</td>
<td>49.4%</td>
<td>N/A</td>
</tr>
<tr>
<td>Baseline RTE5</td>
<td>70%</td>
<td>61.04%</td>
<td>65.21%</td>
<td>0.6333</td>
</tr>
<tr>
<td>Rule-base Test RTE5</td>
<td>73.66%</td>
<td>63.32%</td>
<td>68.1%</td>
<td>0.6483</td>
</tr>
</tbody>
</table>

Table 11: Global ablation test results: the top 2 rows show the F1, recall and precision measures of the system, run on the RTE6 dataset, without the syntax-based resource (baseline) and with it. The bottom two rows show the same baseline vs. rule-base comparison, run on the RTE5 dataset, which also has an accuracy measure.

the overall performance, from 48.45% in the baseline run, to 49.4% in the rule-base test run. In addition, the resource is shown to be more effective on RTE5, improving the baseline F1 result from 65.21% to 68.1%. This result may signify that the style and composition of the dataset itself has a large impact on the usefulness of the resource. All the above differences are statistically significant ($p < 0.02$ using McNeamar test). Performance gains of a few percentage points, are in line with the scale of the positive effects of resources that were typically measured in RTE ablation tests (to illustrate, see the RTE6 results in [Bentivogli et al. 2010a](http://aclweb.org/aclwiki/index.php?title=RTE_Knowledge_Resources)) and the RTE Knowledge Resources page at [http://aclweb.org/aclwiki/index.php?title=RTE_Knowledge_Resources](http://aclweb.org/aclwiki/index.php?title=RTE_Knowledge_Resources).

6 Conclusions

Up to this point, in the main part of my work, I present a comprehensive novel generic syntax-based rule-base for the TE task, based on inputs from previous salient works on the topic, as well as novel contributions. Represented according to the Stanford dependencies standard and based on a well defined formalism, the rule-base is made publicly available for use with TE systems, including full documentation, and tools for rule design and software compilation.

It offers a wide variety of over 60 generic entailment rules, that substitute between copious equivalent or entailing constructions in categories such as: active vs. passive, coordination, apposition, determiners, possessives, and case correction. Also, they can extract simplified IS-A and HAS-A implications from over a dozen generic patterns, and successfully decouples relative clauses. The rules are honed for high resilience to syntactic structure diversity, especially in RTE texts.

Qualitative and quantitative depth evaluations are reported, including a manual dataset analysis, that demonstrate the high potential for knowledge
resources of this type in TE, the wide coverage it has over the required set of syntax-based transformations in this setting, its high accuracy, and how it significantly improves the performance of a concrete entailment engine. Additionally, I show that much work is needed for such systems to make better use of syntax-based resources.

Consequently, I conclude that the resource seems to possess high accuracy and good coverage, and so, at this point, the main research effort should be directed to improving entailment systems, in order to employ it more effectively.

7 A Semantic Truth Value Annotator

7.1 Introduction

This section is structured as a separate part of the thesis, as it reports on an independent research thread, implemented and made publicly available as a different system. It relies on the formalism definitions in Section 3, but can otherwise be read independently of the previous sections.

For many textual inference applications, it is important to know the truth values of the clauses of $T$ (and $H$). For instance, in the following examples, I use the superscripts $ct^+$, $ct^-$ and $ct^?$ to annotate the clause truth values of all head predicates of clauses that are either entailed by the text, contradicted, or whose entailment is unknown:

\begin{align*}
(29) & \quad a. \text{Gal made an attempt}^{ct^+} \text{ to sell}^{ct^?} \text{ her business} \\
     & \quad b. \text{Gal did not make an attempt}^{ct^+} \text{ to sell}^{ct^-} \text{ her business} \\
     & \quad c. \text{Maybe Gal wasn’t smart}^{ct^+} \text{ to sell}^{ct^+} \text{ her business}
\end{align*}

In the above three examples, the sentence-main predicate, represented by the node that dominated the entire parse tree, always has $ct^+$, since every sentence entails itself. Furthermore, in (29a), sell has $ct^?$ because the governing phrasal verb predicate make an attempt to $X$ does not commit the speaker to (does not entail) whether Gal actually sold her business or not. However, in (29b), sell has $ct^-$, because the negated predicate, not make an attempt to $X$, is an implicative construction, which entails the contradiction of $X$. In (29c), sell has $ct^+$, because be smart to $X$ is a factive construction that presupposes that Gal sold her business, regardless of negation. More accurately, presuppositions persist under negation, in questions and if-clauses, while entailments (like in (29a) and (29b)) do not. As these examples demonstrate, factive constructions hold presuppositions, and implicative constructions hold entailments. These predicate types are further defined in Subsection 7.4.1.

Some possible uses for these annotations in the service of natural language inference are:
1. Inferring parts of sentences, like inferring from (29b) that Gal did not sell her business.

2. More precise matching between $T$ and $H$, like detecting that (29a) directly contradicts (29b).

3. Richer and more accurate inference rule matching, being able to match a rule’s LHS against a text not only based on its lemma, part of speech, and syntactic relation features, but also based on semantic annotations. For instance, consider the inference

$$\textit{Ming knew she would} \ X_{\text{manage}} \ \text{to} \ Y \ \Rightarrow \ \textit{Ming knew she would} \ Y$$

where the variable $X_{\text{manage}}$ matches against any verb in the implicative category of \textit{manage}.

The need for this kind of truth value annotation has been addressed in recent work, notably by [Nairn et al. (2006)] and [MacCartney and Manning (2007)], who both developed semantics to compute truth values, and implemented inference systems that utilize them to generate paraphrases and entailments from $T$. Similarly, my main goal is to annotate every clause with truth values (in Subsection 7.4 I distinguish between clausal truth value and predicate truth value), in the same way a parser enriches text representation with syntactic relations. No previous work has focused solely on this challenge, since they discarded their annotations after using them.

In this work, I present TruthTeller, a novel algorithm and a system that focuses on identifying the truth value of each action in the sentence. As input, it takes a parse tree, and generates annotations on each predicate and clause. As a side product, it also annotates negation, and a classification of predicate types (factive, implicative, regular, etc.). See Figure 10 for an illustration of TruthTeller’s output for (29b). In contrast with the cited systems, which focused on generating entailments from the given text, my main objective is to generate useful semantic annotations for all predicates in the parse tree, to be utilized by other applications.

Beyond being the first system to focus on producing comprehensive truth value annotations, my algorithm addresses and combines some aspects of them that have never been addressed in previous work at all, such as phrasal verbs, various syntactic relations like coordination and apposition, generic constructions that alter PREDICATE TYPE (e.g., turn a factive into an implicative), explicit annotation of UNKNOWN truth value, and generic presuppositional constructions, like relative clauses. Other novelties of the system are that it is open source, made publicly available as a tool within the BiuTee release [Stern and Dagan (2011)] and that it offers a relatively simple algebraic model for the recursive truth value computation, and is

\[\text{http://www.cs.biu.ac.il/~nlp/downloads}\]

46
Figure 10: An example of a parse tree with semantic annotations. Each edge holds a Stanford dependency relation, and each node contains a lemma and Penn POS, separated by a slash. These three features are added by the parser. In addition, every predicate node is annotated with four more semantic features (explained in Subsection 7.4), listed from left to right: predicate implication signature, negation and uncertainty, clause truth value, and predicate truth value. The value P stands for “positive”, N for “negative”, and U for “uncertain” or “unknown”.

\[ \text{Gal did not make an attempt to sell her business.} \]
accompanied by a publicly available lexicon of over 1,700 implicative and factive predicates.

Finally, I note that the current work adopts the Stanford dependency representation standard for parse trees (like the generic syntax-based rules presented in Section 4), and is intended as an opening point for future extensions. For instance, as mentioned, it can potentially employ more annotation types, identify a wider variety of constructions that trigger annotations, and incorporate many other types of lexical resources, beyond the existing single word and phrasal verb lexicons (discussed below), such as factive phrases (make pretence to X), phrases of verbs with adjectives (see it fit to X) etc.

This section is structured as follows: Subsection 7.2 discusses previous work; Subsection 7.3 lays out the formalism for my annotation features and annotation rules; Subsection 7.4 specifies the four semantic annotation types used; Subsection 7.5 presents the algorithm, constructed as a hybrid of local pattern-matching annotation rules and a recursive truth value calculus; Subsection ?? presents some implemented and some potential uses for my annotations in the service of TE. Subsection 7.6 reports a qualitative precision evaluation, as well as a quantitative ablation test; and finally, Section 7.7 offers some conclusions.

7.2 Previous Work

The ground work on much of the theory in this study was laid out in the following two highly acclaimed publications. First, Kiparsky and Kiparsky (1970) established the presuppositional qualities of factive verbs and other, more generic, constructions. They also highlight how too-and enough-constructions effectively change a factive into an implicative, e.g., in example (29a) be smart to X presupposes, while be smart enough to X entails. Second, Karttunen (1971) focused on the entailment qualities of (six categories of) implicative verbs. Of late, Karttunen (2012) revisited and expanded on his past work, producing a novel publicly available lexicon of about 300 implicative phrasal verbs. All these commonplace patterns are implemented as annotation rules (defined below) in Subsection 7.5.

In the more recent thread of empirical work, Nairn et al. (2006) (together with Karttunen), were the first to define partial computational semantics for simple implicative and factive verb constructions, and the interaction between them. Though it was not specified, their work can be considered an implementation of the “natural logic” task, a term first coined by Lakoff (1972). To this end they compiled a novel lexicon that classifies a large set of single-word complement-taking factive and implicative verbs. Additionally, the authors presented a design and a working system that computes the “relative polarity” (i.e., truth value) of predicates, as a top down recursive calculation, from sentence main predicate to deepest embedded clauses.

Bar-Haim et al. (2007) implemented parts of Nairn et al.’s work in their
own RTE system, in the form of **annotation rules**, that mark polarity values in-situ (without recursion) in a sentence, based on matching simple generic patterns of predicate-argument constructions and negation words. Similar rules serve below as building blocks for most parts of **TruthTeller** (see Section 7.3.2).

MacCartney and Manning (2007, 2009) proposed and implemented a new model of “natural logic” which, similar in spirit to this study, characterizes formal semantic inference patterns in terms of syntactic templates, which are as close as possible to natural surface forms. Thus, the authors hoped to gain from both the accuracy of formal semantic calculus, along with the robustness of text based systems. This model extended a bottom-up recursive monotonicity calculus to incorporate semantic exclusion, and partly unified it with Nairn et al.’s account of implicatives and factives. They first defined an inventory of semantic binary relations, which includes representations of both containment and exclusion. Then they described a general method for establishing the overall semantic relation between $T$ and $H$: given a sequence of atomic edits (that add, delete or substitute words) which transforms $T$ into $H$, they determined the semantic relation generated by each edit: first, project each semantic relation into an atomic relation, according to the context in which the edit occurs; then join atomic semantic relations across the edit sequence. They present an implemented system based on this model in (MacCartney et al, 2008; MacCartney and Manning, 2009; MacCartney, 2009).

Beside their pioneering achievements in the field of natural logic, the above systems shared a few noteworthy shortcomings. For one, they did not combine the calculus of factives and impicatives with other generic constructions that:

a) alter **predicate signature**, e.g., in

   i. *Maybe Gal wasn’t smart*$_{t^+}$ *to sell*$_{t^+}$ *her business*

   ii. *Gal wasn’t smart*$_{t^+}$ *enough to sell*$_{t^-}$ *her business*

   the predicate *be smart to X* is factive in (i), but is implicative (sensitive to negation and modality) in (ii), because of the adverb *enough*. Apparently, other systems would categorically classify each predicate either one way or another.

b) invoke an explicit annotation of **unknown** truth values, like in *The driver might beat*$_{t^?}$ *traffic if he hurries*, where it is unknown if the driver either hurried or beat traffic. This annotation helps distinguish cases where a hypothesis can be neither proven nor contradicted from the text.

c) identify **generic presuppositional constructions**, like many relative clauses, e.g., *I passed by the school which Josh used*$_{t^+}$ *to attend*
d) copy or propagate truth values via relations other than strict verbal predicate-complement relations, such as nominal predicate-complement, adjectival predicate-complement, coordination and apposition.

In addition, they had, what Nairn et al (2006) called, lexicographic gaps, meaning that the identification of predicate-complement relations in natural texts is brittle, especially in collocations and nominals like take the trouble to, see fit to, receive confirmation about. And without a robust syntactic abstraction of a compound natural-language sentence into a few simple chains of predicate-complement relations, no higher level truth value calculus can take place. Finally, these systems were not made publicly available.

In contrast, TruthTeller covers all the noted capacities of the aforementioned systems, and extends them to address the missing features laid out above. For instance, it is the first to be an open source tool, allowing other groups to modify its existing annotations, and add to them. In addition, my algebra for recursive truth value computation is simpler and more concise than in previous work. I also report a manual evaluation of the precision of my annotations, as well as an ablation test of TruthTeller, as a preprocess component within a public domain state-of-the-art textual entailment system, BiuTee (Stern and Dagan 2011).

The last referenced system was reported by Mausam et al (2012), who presented a novel Open Information Extraction system, that not only extracts relation phrases and associated arguments in arbitrary sentences (which is common for IE systems), but, in cases where the assertion is found to be non factual, attaches an extra attribute to the extracted relation that is supposed to give the user a description of its uncertainty. This way, given the sentence

*Early astronomers believed that the earth is the center of the universe*

their system extracts the subject-predicate-complement tuple (the earth; be the center of; the universe), and attaches to it the attribute (AttributeTo believe; Early astronomers). A number of generic conditional structures also generate similar attributes (if, when, although, because,...). All the above attributes are identified by a closed set of dependency parse tree templates and a token lexicon of uncertainty verbs (believe, doubt, suggest, etc.), that seem to work like our annotation rules.

Although these attributes describe uncertainty with more granularity than our system does, by distinguishing between actions that were believed, doubted, suggested etc., it is left unclear how a client application could benefit from it (at least, in the domain of Textual Entailment). In contrast, TruthTeller’s annotations cover a much broader range of constructions that convey uncertainty, using a much larger lexicon (see below), and outputs its classifications with high accuracy, in a straightforward format (where each predicate is annotated as positive, negative or neutral truth value).
Additional effort was put into constructing a lexicon of single word implicative and factive predicates, by far the largest of its kind, containing finer classifications (explained in Subsection 7.4), and the first such resource to be made publicly available. It was formed by manual classification of about 300 such predicates, and expanding the collection semi automatically with direct synonyms and entailments from Wordnet (Fellbaum 1998; Miller 1995), to over 1,700. I utilize it alongside the lexicon of phrasal verbs made by Karttunen (2012), which has about 300 entries of its own.

TRUTHTELLER is designed for robustness, coping with a significant degree of language variation around each supported generic construction, which are specific to the Stanford dependency standard as produced by the EasyFirst parser. This was achieved through extensive error analysis and tweaking the annotation rules (discussed in Subsection 7.5). As a last point, to the best of my knowledge, TRUTHTELLER is the first publicly available system to directly address natural logic computations and to annotate all clauses of a text with truth values.

On the other hand, it does not yet combine quantifiers (except for a few cases involving negation words) nor monotonicity in its calculus (though it does address connectives), like MacCartney and Manning (2009) do. Monotonicity accounts for inference patterns such as:

\begin{align*}
\text{(30)}\quad &\text{a. tall boy} \Rightarrow \text{tall person} \\
&\text{b. not a tall boy} \Leftarrow \text{not a tall person}
\end{align*}

My annotation scheme cannot independently detect these two inferences, because they rely on identifying the hyponymy between two phrases, be it lexically e.g., boy \(\rightarrow\) person, or syntactically e.g., boy with cap \(\rightarrow\) boy. The inferences also rely on identifying that one expression only permits substitution with more general noun phrases (“up monotonicity”), while the second permits the opposite (“down monotonicity”).

These phenomena fall out of the scope of the study, because they do not impact the truth values of the constituents of the original sentence, which are my main focus, but only affect the entailment of novel sentences, that are variants of the original. These and other capabilities can be added to my system in the future.

The system’s algorithm can be divided into 2 distinct parts that work in sequence. First, based on an extension of the formalism for annotation rules from Bar-Haim et al (2007), including a few extensions of my own, it applies a series of rules to the text (see Subsection 7.3.2) in order to annotate every semantic quality that can be identified based on local constructions. Second, extending the above-mentioned recursive systems, it runs a novel top-down recursive process for computing the truth values of all clauses.
7.3 Formalism of Annotation Features and Annotation Rules

This subsection describes the assumed formalism for annotations, and for annotation rules that apply annotations on parse trees, following the formalism in Bar-Haim et al. (2007), with a few additions, detailed below.

7.3.1 Annotation Features

I wish to modify the formalism for entailment rules in Section 3, to design a mechanism for adding lexical-semantic features to parsed text, called annotation rules. In some cases, I want to add a feature relevant to a single word, like when marking negation words. In other cases, the feature describes an entire phrase or clause (a subtree), like when computing the entailment judgement (positive or negative) of a sentence. The interpreted scope of an annotation depends on its definition. These annotations are implemented as auxiliary features in parse tree nodes, besides the original features, generated by the parser, like the lemma and POS tag.

Since annotation features are technically no different than other features in the node, it is straightforward to use them in the L-matching step of entailment rules (see examples in Section ??). Furthermore, like other node features, annotations may be altered by many means. However, in this work, I mainly focus on the mechanism of annotation rules, as well as the recursive annotation function in Subsection 7.5.3 (which cannot be implemented with rules).

7.3.2 Annotation Rules

Much like entailment rules were used to represent lexical syntactic transformations, annotation rules serve to represent lexical-semantic feature tagging, such as negation, classification of verbs or truth value. They perform this by adding, removing and modifying annotation features in the parse tree of $T$.

Technically, annotation rules are similar to entailment rules, in that both have a left hand side (LHS) $L$, which matches against a certain pattern in $T$. In contrast, annotation rules do not have a right hand side. Rather, each node of $L$ may hold an attached set of annotation substitution parameters, containing the information needed for replacing some or all of the node’s annotations. A substitution set has up to one parameter for each annotation type (defined below), whose value is either an explicit annotation value, which replaces its corresponding value in the node, or is a special flip value, which “flips” the corresponding value in the node from positive to negative, or vice versa. The flip operation is my only significant extension to the formalism of Bar-Haim et al. (2007), and is defined only for substitution parameters of those annotation types that take positive and negative values.
7.4 Annotation Types

This subsection defines the specific annotation features I employ in this work, while the next subsection presents the algorithm by which the system computes them.

I employ four lexical-semantic features:

1. Predicate Implication Signature ($\text{sig}$)
2. Negation and Uncertainty (NU)
3. Clause-Truth (CT)
4. Predicate Truth (PT)

Below, I define each type of feature, followed by an illustrative example.

Technically, any parse tree node can hold any annotation. Nonetheless, my annotations are meaningful and well defined only in the context of parse tree nodes that are predicates, be them verbs, nouns (inc. pronouns) or adjectives. While virtually all verbs are predicates, I consider nouns to be predicative only in copular constructions, and in instances where they have complements, i.e., where it is visibly clear the noun is a nominalization. To illustrate, I consider the boldface nouns in (31a) and (31b) as predicates, in contrast to the nouns not emphasized.

(31) (a) My next guest is a famous writer and publisher
    (b) The chief noted recent successes in filing more than 100 charges
        for people not wearing seatbelts
    (c) All the leaves are brown and the sky is gray
<table>
<thead>
<tr>
<th>Category</th>
<th>Sig</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implicative 1</td>
<td>+/-</td>
<td>Gal managed to escape ⇒ Gal escaped&lt;br&gt;Gal didn’t manage to escape ⇒ Gal didn’t escape</td>
</tr>
<tr>
<td>Implicative 2</td>
<td>+/?</td>
<td>Gal was forced to sell ⇒ Gal sold&lt;br&gt;Gal wasn’t forced to sell ⇒ no entailments</td>
</tr>
<tr>
<td>Implicative 3</td>
<td>?/-</td>
<td>Gal was permitted to leave ⇒ no entailments&lt;br&gt;Gal wasn’t permitted to leave ⇒ Gal didn’t leave</td>
</tr>
<tr>
<td>Implicative 4</td>
<td>-/+</td>
<td>Gal forgot to pay ⇒ Gal didn’t pay&lt;br&gt;Gal didn’t forget to pay ⇒ Gal paid</td>
</tr>
<tr>
<td>Implicative 5</td>
<td>-/?</td>
<td>Gal refused to fight ⇒ Gal didn’t fight&lt;br&gt;Gal didn’t refuse to fight ⇒ no entailments</td>
</tr>
<tr>
<td>Implicative 6</td>
<td>?/+</td>
<td>Gal hesitated to ask ⇒ no entailments&lt;br&gt;Gal didn’t hesitate to ask ⇒ Gal asked</td>
</tr>
<tr>
<td>Factive</td>
<td>+/-</td>
<td>Gal was glad to come ⇒ Gal came&lt;br&gt;Gal wasn’t glad to come ⇒ Gal came</td>
</tr>
<tr>
<td>Counter</td>
<td>-/-</td>
<td>Gal pretended to like her ⇒ Gal didn’t like her&lt;br&gt;Gal didn’t pretend to like her ⇒ Gal didn’t like her</td>
</tr>
<tr>
<td>Regular</td>
<td>?/?</td>
<td>Gal wanted to fly ⇒ no entailments&lt;br&gt;Gal didn’t want to fly ⇒ no entailments</td>
</tr>
</tbody>
</table>

Table 12: Implication signatures. Based on the accounts in MacCartney and Manning (2009) and Karttunen (2012).

Finally, adjectives are considered predicates only in copular clauses, like in (31c).

7.4.1 Predicate Implication Signature

The signature of a predicate $p$ (denoted by $\text{sig}(p)$) represents a lexical-semantic categorization of predicates over (6 classes of) implicatives, (normal and counter) factives and regular predicates. Its nine possible values are shown in the Sig column in Table 12 following MacCartney and Manning (2009) and Karttunen (2012).

Each signature has a left sign and a right sign. The left sign determines the truth value of the complement when the predicate is in positive contexts (e.g., not negated, as explained under “predicate truth” below), while the right sign applies in negative contexts. Each sign can be either $p$(ositive), $n$(egative) or $u$(nknown), the latter signifying that no entailments are available.

As an example, I use the notation $-/+$ for the fourth signature to indicate that $\text{forget to } X$ yields a negative entailment for $X$ in a positive context, with the $N$ on the left, and a positive entailment in a negative context, with the $P$ on the right. For example:
• Gal forgot to lock the door ⇒ Gal didn’t lock the door

• Gal didn’t forget to lock the door ⇒ Gal locked the door

The same holds for all other predicates with this signature. Similarly, the second signature +/? indicates that constructions like force to X yield a positive entailment for X in a positive context, and an unknown entailment for X in a negative context, i.e., not forcing X does not say anything about whether X was done or not. As for factives with the +/+ (-/-) signature, their arguments are positively (negatively) entailed in all contexts. All non implicative and non factive predicates are considered regular ?,/?, which means they yield an unknown entailment for their arguments in all contexts.

It is important to note, that each predicate may have different signatures, depending on the type of its complements, i.e., non-finite (infinitive and gerundive clauses) vs. finite (tensed clauses and noun phrases). For instance, the verb forget is implicative with non-finite arguments, but is factive with finites. Compare:

• Gal forgot to lock the door ⇒ Gal did not lock the door

• Gal forgot that she locked the door ⇒ Gal locked the door

This additional layer of complexity is addressed by implementing a set of signatures that is finer than in previous related works. For instance, forget actually does not get the signature +/-, nor +/+; but “+/+FIN_/+/-NONF”, to represent its duality. These details are concealed throughout this paper, and can be found in TruthTeller’s documentation.

7.4.2 Negation and Uncertainty

NU (also noted below as nu(p)) combines two properties of a predicate that can be identified by the presence of local modifiers: whether the speaker casts doubt on the action or event, and, if not, whether it is negated. Typical local modifiers that express UNCERTAINTY are maybe, perhaps, should and may, while negation modifiers include not, never and neither.

Notice that according to my definition, uncertainty overrides negation, and the sentence:

Rajaa might not make it

events uncertainty and is not negated. Other forms of contradicting and casting doubt on a predicate are expressed in CT and PT, to follow.

Correspondingly, NU takes these three values: {NU-, NU?, NU+}. NU+ describes predicates that are neither negated nor uncertain; NU- those that are negated but have no uncertainty modifiers; and NU? those that have uncertainty modifiers (which override the effect of negation). See the NU annotations in example (32) below.
7.4.3 Clause Truth

CT (noted as $ct(p)$) corresponds to the notion of polarity in Nairn et al (2006). It represents whether the clause headed by the predicate is entailed by the complete sentence, negatively entailed or unknown. The latter is used for cases where the system either identifies that the speaker makes no commitments about the veridity of the clause, or where the system is unable to make any judgement at all. Correspondingly, CT takes these three values: \{CT+, CT-, CT?\}.

7.4.4 Predicate Truth

The last annotation type is PT (denoted as $pt(p)$), and it can be concisely defined as the binary product of NU and CT (accurately defined in Subsection 7.5.3):

\[
PT = NU \cdot CT
\]

It represents whether the action described by the predicate is entailed from the text. It is similar to CT, but differs in the sense that CT captures the entailment of the entire clause, while PT focuses solely on the predicate, without its arguments. It takes analogous values: \{PT+, PT-, PT?\}.

The above annotations are illustrated in the following sentences:

\[(32)\]  
\[a.\] Motorcycles are not allowed ?/-,nu-,ct+,pt- to park ?/?,nu+,ct-,pt- here  
\[b.\] The order might not arrive ?/?,nu?,ct+,pt? on time

In (32a) the adjective allowed is a ?/- implicative, it is negated by not, gets CT+ since the clause it heads is the main clause of the sentence (and every text entails itself), and it gets PT- since the action of being allowed does not occur. Also, the verb park is regular ?/?, is not negated, its local clause is negatively entailed as the argument of not allowed to, and correspondingly the action of parking does not occur. In (32b), arrive is a regular verb, has uncertain negation because of might, its clause is self entailed (like allowed), and it gets PT? because the speaker is uncertain about the arrival (again, because of might).

7.5 Annotation Algorithm

This section describes the algorithm for generating annotations on parsed sentences. It is comprised of the following main steps:

1. Annotating implication signatures (Subsection 7.5.1)  
   (a) Single word predicates
(b) Phrasal verbs
(c) Generic constructions that alter signatures

2. Negation and uncertainty (NU) annotation (Subsection 7.5.2)
   (a) Positive NU - by default
   (b) Negative NU - predicates modified by negation expressions
   (c) Uncertain NU - predicates modified by indefinite pronouns, uncertain auxiliaries and uncertain adverbs
   (d) NU propagation to conjuncts

3. Clause Truth (CT) and Predicate Truth (PT) annotation (Subsection 7.5.3)
   (a) Generic presupposition constructions - predicates with adverbial clauses, WH arguments
   (b) CT and PT recursive computation
      i. (Applied in tandem with the recursive computation) generic presupposition constructions - relative clauses and participle clauses

All steps are implemented through the sequential application of annotation rules, as defined in Subsection 7.3.1, with two exceptions: the use of the signature lexicons in steps (1a) and (1b) and the recursive CT and PT computation of step (3b). These exceptions are explicitly highlighted and discussed in the corresponding subsections.

As a general note, behind the structures addressed by each annotation rule, there is a great deal of language diversity to be accounted for. Because this complicates the annotation rules, the finer details of many rules are concealed here, and may be found in the download package. Furthermore, naturally many of these structures cannot be accurately captured by rules within my formalism. Therefore several rules carry caveats (i.e., may produce incorrect annotations), but my system evaluations on RTE datasets show that in general all rules are beneficial.

7.5.1 Predicate Implication Signatures

Signatures are annotated in this step and later serve mainly in computing CT (Subsection 7.5.3).

**Signature lexicons** My system identifies the signature of implicatives and factives using two lexicons, one for single-word predicates and one for phrasal verbs. My single-word lexicon is similar to those used in [Nairn et al 2006] and [Bar-Haim et al 2007], but is far greater, holding over 1,700 pairs
of predicate and signature. The signatures of the lexicon are those listed in Table 12.

In particular, the +/? signature serves as the category for all predicates in T not listed in my lexicons. This choice follows from the observation that most verbs in a positive context entail their objects, be it the existence of their nominal objects, or the proper entailment of their clausal objects, e.g.:

- Sheela broke a plate again ⇒ there was a plate
- The kids played with a ball that someone had left here ⇒ there was a ball, and someone had left it here

The annotation precision, evaluation reported in Subsection 7.6.1 supports this choice.

The lexicon was built semi automatically, out of a kernel of approximately 300 manually inspected predicates, which was expanded by adding WordNet synonyms [Fellbaum (1998); Miller (1995)] to each category. I observed that substituting a manually classified predicate with its immediate synonym usually preserves the signature, and preliminary ablation tests show that the BiUTE inference system performs slightly better with the expanded lexicon. Nonetheless, due to the semi-automatic construction process, the lexicon may contain few errors.

The second lexicon is the implicative phrasal verb lexicon of Karttunen (2012), adapted into my framework. Each phrasal verb consists of a verb and a noun, such as take the time. The two lexicons are made available as part of the download package.

**Signature Annotation** The annotation rules for signature have some extra features not shared by the other rules, enabling them to lookup predicates in the lexicons. Furthermore, while annotating a single word predicate, represented by a single node, is trivially performed through a simple lexicon lookup, annotating phrasal verbs involves syntactic constructions much more varied, complex and delicate.

In order to conform with the rest of the algorithm, all annotations relevant to a phrasal verbs must be kept at the parse node of the verb, which is invariably the root of the phrasal verb’s subtree. This need raises several challenges, because phrasal verbs span over two or more nodes, and therefore can appear in a variety of syntactic constructions, including different coordination and predicate-complement relations. To this end, I manually characterised and formulated most of these phrasal verb constructions into reliable annotation rules, which correctly identify the verb and the noun, and tag the verb’s signature via lexicon lookup.

Consequently, for each one of the above-mentioned generic predicate-complement constructions, I composed an annotation rule schema (like the schemas used in Section 4) that matches against the construction, and, in
Figure 12: Generic schema of a signature annotation rule, matching against any verb+noun phrasal verb, where the noun is in a coordination, e.g., *take the risk and the chance*. The parameters for the phrasal verb’s lemmas in the LHS and new signature in the RHS hold unique place-holder strings that are replaced with entries from the phrasal verb lexicon, at compilation time.

addition, has unique place-holder values in the nodes where the predicate’s lemma(s) and new signature are supposed to be, see example in Figure 12. At compilation, the schema is not only compiled into formal rule(s) as usual, but a copy of the rule is created for each predicate-signature entry in the appropriate lexicon (either single word or phrasal verb), and the place-holders are replaced with the entry’s values. This special function of the compiler is activated by the presence of the special place-holder values in the schema.

Abstracting the above-mentioned implementation details, the signature annotation process is described in the following rule:

**R1.1:** Annotate each predicate with its signature, according to the single-word and phrasal verb lexicons. Unclassified predicates get +/? by default.

(33) *Ruth regrets leaving school* ⇒ *Ruth regrets*+/+ *leaving*/? *school*

(34) *Gil made an effort to show up* ⇒ *Gil made*/− *an effort to show*/? *up*

**Generic structures that alter signatures** After the initial signature annotation, the following two rules correct signatures where necessary, according to two generic constructions.

Kiparsky and Kiparsky (1970) noted that most +/+ factive predicates, when modified by *enough* and *too* adverbs, behave like +/− and −/+ implicatives, respectively. Compare:

(35) a. *John was not clever to leave early - though he did*

b. *John was not clever enough to leave early - so he didn’t*
In the first example, *clever* behaves like a +/+ factive which presupposes its argument, while, in the second, it is modified by *enough* and behaves like a +/- implicative, which entails its argument in accordance with its negation. The pattern is captured by this set of rules:

**R1.2:** +/+ factive predicates in *enough* and *too* constructions are annotated +/- and -/+ correspondingly.

The list of *enough* adverbs includes: *enough, sufficiently, satisfactorily, aptly, adequately*.

The list of *too* adverbs includes: *too, excessively*.

Figure 13 shows the schema addressing the *-too* constructions.

The next two sentences exemplify another similar observation, that most verbs in the passive voice and with the *into* preposition, behave like +/? implicatives.

(36)  
   a. The workers **were pushed into** signing the deal (so they had to sign)  
   b. The workers **were not pushed into** signing the deal (but maybe they did)

This phenomenon was also noted by (Karttunen, 2012), and is captured in the following rule:

**R1.3:** All verbs in the *passive+into* construction are annotated +/?

### 7.5.2 Negation and Uncertainty Annotation (NU)

NU is detected by applying the following annotation rules, in order, on every predicate: Basically, they identify a closed set of negation modifiers and uncertainty modifiers around the predicate. It later serves in the computation of CT and PT.

**R2.1:** Initially, all predicates are annotated NU+
(37) This man was seen at the site ⇒ This man was seen^{nu+} at the site

R2.2: Predicates in the following structures flip their NU, from nu^+ to nu^−, and vice versa:

(a) modified by one of the following negation adverbs and determiners: no, not, n’t, ain’t, nor, neither, never, almost, nearly

(38) Reporters were not seen^{nu+} at the site ⇒ Reporters were not seen^{nu−} at the site

(b) are one of the following nouns: none, neither, nobody, nothing, nowhere

(39) Nobody^{nu+} was seen at the site ⇒ Nobody^{nu−} was seen at the site

(c) whose subject has nu^− (feeding on the previous rules)

(40) Nobody^{nu−} was seen^{nu+} at the site ⇒ Nobody^{nu−} was seen^{nu−} at the site

R2.3: Predicates, whose object has nu^− are annotated nu^− (feeding on the previous rules)

(41) Reporters were seen^{nu+} at neither site^{nu−} ⇒ Reporters were seen^{nu−} at neither site^{nu−}

R2.2 successfully captures many cases of double negation, where two negations cancel each other out. Compare this example, annotated by two applications of R2.2a (matched against almost and no), with (40):

(42) Almost nobody^{nu+} was seen at the site
    ‡ (R2.2a)
    Almost nobody^{nu−} was seen at the site
    ‡ (R2.2b)
    Almost nobody^{nu+} was seen at the site
    (R2.2c doesn’t match)

Double negation is an example for cases where the FLIP value is necessary, instead of using explicit feature values in the substitution annotations set.

Note that R2.3 uses explicit annotation substitution values (i.e., only changes nu^+ to nu^−), where R2.3 uses the FLIP operation, because it matches against constructions of object negation, which is not cancelled by subject negation nor by predicate negation. Such sentences with both object negation and another negation are found mostly in colloquial English. Consider, for instance:

(43) Nobody was seen^{nu−} at no site
Figure 14: Schema for flipping the NU of predicates, whose nominal subject was negated, by previous rule applications. The \texttt{ruleType=FLIP\_ANNOTATION} label indicates this is a flip annotation rule, while the \texttt{flip\_nu} label in the top RHS node specifies which annotation to flip. Notice the subject must be nominal, \texttt{NSUBJ\_NSUBJPASS}, and not clausal, \texttt{CSUBJ\_CSUBJPASS}, because negated clausal subjects generally do not negate their governors, e.g., \textit{That you didn’t tell me about it, is so upsetting}.

and compare it with the effect of double negation in (42).

\textsc{TruthTeller} also addresses more complex negation constructions, like combinations of three or more negation modifiers, though I have noticed it commits relatively many errors on them. Nonetheless, such sentences are scarce in the wild, and are unlikely to have a real effect on performance.

The next set of rules addresses constructions that convey uncertainty:

\textbf{R2.4:} Predicates in the following structures are annotated \texttt{NU}?

(a) modified by these uncertainty auxiliaries: \textit{should, ought, could, ought to, may, might, would}

\begin{align*}
44) & \text{Pluto} \text{ should} \text{ be carefully observed}^{u+} \Rightarrow \text{Pluto} \text{ should} \text{ be carefully observed}^{努？}
\end{align*}

(b) modified by these uncertain adverbs: \textit{perhaps, possibly, maybe, probably, theoretically, presumably, potentially, hopefully, seemingly, apparently, likely}

\begin{align*}
45) & \text{Perhaps Pluto will not be downgraded}^{努−} \text{ from planet to object} \Rightarrow \text{Perhaps Pluto will not be} \text{ downgraded}^{努？} \text{ from planet to object}
\end{align*}

(c) in clauses headed by these indefinite relative pronouns: \textit{whoever, whomever, whichever, whatever, wherever, whenever}

\begin{align*}
46) & \text{Tell my story to whomever will listen}^{努+} \Rightarrow \text{Tell my story to whomever will listen}^{努？}
\end{align*}

(d) are the main predicate of a question, identified by a question mark

\begin{align*}
47) & \text{Did you actually visit}^{努+} \text{ the Taj Mahal?} \Rightarrow \text{Did you actually visit}^{努？} \text{ the Taj Mahal?}
\end{align*}
In the case of coordination (conjunction) structures, R2.2 and R2.3 will not create accurate NU annotations on conjuncts other than the first, due to the way coordinations are represented in the Stanford dependency standard. According to the standard, only the first conjunct is connected to the rest of the tree, and is modified by all other conjuncts (see more detailed discussion in Subsection 4.3). Therefore, after applying the above rules, the following two rules are necessary to correctly propagate NU to conjuncts:

R2.5: Propagate nu- to conjuncts that have nu+ (from the first conjunct to the others)

\[(48) \text{Sara doesn’t smoke}^{\text{nu}-} \text{nor drink}^{\text{nu}+} \text{ much} \Rightarrow \text{Sara doesn’t smoke}^{\text{nu}-} \text{nor drink}^{\text{nu}-} \text{ much}\]

R2.6: Propagate nu? to all conjuncts (from the first conjunct to the others)

\[(49) \text{Perhaps Sara doesn’t smoke}^{\text{nu}?} \text{ nor drink}^{\text{nu}-} \text{ much} \Rightarrow \text{Sara doesn’t smoke}^{\text{nu}?} \text{ nor drink}^{\text{nu}-} \text{ much}\]

The last two rule schemes are illustrated in Figure 15. Notice the subtle difference between them: the first only changes nu+ (to nu-), while the second changes either nu+ or nu- (to nu?). This feature, along with the fact that all the above nu? annotation rules are applied after the nu+ and nu- rules, gives precedence to the nu? annotation over nu-. This ordering reflects the semantic precedence of uncertainty over negation, which is why I take (49) to mean that Sara may drink, instead of meaning that she does not drink.

### 7.5.3 Clause Truth and Predicate Truth

CT is determined either locally, via annotation rules that identify generic presupposition structures, or via a recursive computation, following the procedure defined in Nairn et al. (2006) and MacCartney and Manning (2009), within the “natural logic” framework. As an additional product of the recursive procedure, PT is computed for every predicate, as a function of the local NU and CT (defined below).

#### CT presupposition annotation rules

The next set of rules identifies a wide range of generic presupposition structures, based on syntactic structures and a closed set of prepositions and pronouns.

R3.1: Generic presupposition - the following constructions presuppose their main predicate, annotating it with CT+

\[(a) \text{Sentence main predicate (i.e., a sentence presupposes itself)}\]
Figure 15: Top: schema for copying NU- to conjuncts in a coordination structure. Bottom: schema for copying NU? to conjuncts, which overrides the effect of the first schema in cases where the two apply to the same coordination. The NU on the first conjunct is annotated by other rules that precede these two.

(50) Every sentence is true in a sense ⇒ Every sentence is true\textsuperscript{c}\textsubscript{t} in a sense

(b) Parataxis - the parataxis relation (from Greek for “place side by side”) is a relation marked by the parser between the main verb of a clause and other sentential elements, such as a sentential parenthetical, or a clause after a “;” or a “;\textsuperscript{11}

(51) The guy, John said, left early in the morning ⇒ The guy, John said\textsuperscript{c}\textsubscript{t}, left early in the morning

(c) Definite noun phrase, identifiable by
   i. Definite articles, including: the, this, that, these, those, either, some
   ii. Possessors, e.g., Her/Helen’s creativity\textsuperscript{c}\textsubscript{t} knows no boundaries
   iii. Proper names, e.g., I met a certain Jack\textsuperscript{c}\textsubscript{t}, who claims he knows you
   iv. Pronouns (the specific case of WH pronouns poses a caveat to this rule)

(d) Presupposing adverbial clause - any clause opened by one of these phrases: before, during, after, while, as (while), whereas, when, where, why, how, how come, because, since, due to, owing to,

\textsuperscript{11}Taken from the Stanford typed dependencies manual (de Marneffe & Manning, 2008)}
Figure 16: Schema defining various WH-complement annotation rules. The WH can be any of who, who, what and which, and must modify its parent predicate as a nominal subject, passive subjent, or object.

following, than, though, although, despite, in spite of, besides, yet, therefore and like

(52) He sleeps after he eats, while the kids play outside ⇒ He sleeps after he eats\textsuperscript{ct+}, while the kids play\textsuperscript{ct+} outside

(e) WH argument - any clause headed by a WH-phrase that is an argument (subject or object): who, which, whom and what

(whether seems to be the only WH word that has no presuppositional qualities)

(53) The police need to know where he was last seen ⇒ The police need to know where he was last seen\textsuperscript{ct+}

Figure 16 illustrates the schema for the WH argument rules in (R3.1e).

CT and PT recursive calculation So far I have described how the annotations may be applied via annotation rules. While my formalism of rules covers a wide variety of local phenomena, pertaining to a constant-size subtree in the text, it is unsuitable for capturing long-distance recursive calculations. For this reason, the following recursive procedure, for annotating CT and PT, is used: starting with a predicate that was already assigned a CT value (typically the sentence main predicate), PT is calculated as the product of its NU and CT, as defined in Table 13, then the CT of (each of) its complement(s) is computed as the product of the predicate’s signature and PT, as defined in Table 14. For example, in (54), where the main predicate is the phrasal verb showed hesitation, its NU- and CT+ determine the PT-, and in turn, the ?/+ signature and the PT- determine the complement’s CT+.

(54) The newbe showed\textsuperscript{?/+},nu-\textsuperscript{-},ct+\textsuperscript{+},pt-\textsuperscript{-} no hesitation to quit\textsuperscript{ct+} while she was ahead
Table 13: Definition of \( pt(p) \), the predicate-truth of \( p \), as a function of \( nu(p) \) and \( ct(p) \). The values in row 3 and column 3, where the predicate has \( nu? \) (e.g., *maybe*, *possibly*), or \( ct? \) (e.g., *think that X*), are all \( pt? \), because uncertainty always takes precedence over effects of positive or negative entailment. Otherwise, PT is the simple binary product of positive and negative signs.

<table>
<thead>
<tr>
<th>NU / CT</th>
<th>CT+</th>
<th>CT-</th>
<th>CT?</th>
</tr>
</thead>
<tbody>
<tr>
<td>NU+</td>
<td>PT+</td>
<td>PT-</td>
<td>PT?</td>
</tr>
<tr>
<td>NU-</td>
<td>PT-</td>
<td>PT+</td>
<td>PT?</td>
</tr>
<tr>
<td>NU?</td>
<td>PT?</td>
<td>PT?</td>
<td>PT?</td>
</tr>
</tbody>
</table>

Table 14: Definition of \( compCT(p) \), the CT of a complement predicate \( p \), as a combination of its parent predicate’s signature and PT. \( compCT(p) \) is defined only for predicates, which are complements of other predicates.

This step reoccurs in each complement, proceeding from top to bottom, through all predicates, until reaching the leaves of the tree.

Notice Table 14 includes all signatures from Table 12: \(+/-\) factives presuppose their arguments by setting \( ct+ \) (\( ct- \)), no matter the context, while \(?/\) predicates always imply \( ct? \). In contrast, the entailments of implicatives are sensitive to PT.

These definitions are not only consistent with the more loose definition of PT initially given in Subsection 7.4.4 but they accurately define the “context” by which to interpret signatures (mentioned there): where an implicative has \( pt+ \), the CT of its complement matches the left sign in its signature (as shown by the entailments in the odd examples of Table 12), and where \( PT = pt- \), the CT of the complement matches the right sign (as in the even examples of Table 12). Where \( PT = pt? \) the result is \( ct? \) for every implicative signature. For instance, in (54) *show hesitation* is \( pt- \), thus the right side of its \(?/+\) signature determines the \( ct+ \) annotation of
its complement *quit*.

To complete the picture, the following recursive formula is used to for calculating CT (and PT) from the sentence main predicate (and other presupposed predicates), down to the most deeply embedded predicates:

\[
ct(p) = \begin{cases} 
ct^+ : & p \text{ is presupposed according to a rule} \\
compCT(p) : & \text{otherwise, and } p \text{ is a complement} \\
t^? : & \text{otherwise (default)}
\end{cases}
\]

With this definition, I expand on the cited research on natural logic, to integrate phenomena like generic and lexical presupposition and uncertain entailment into the computation. Significantly, CT values from annotation rules (based on local generic presupposition) are given precedence over the recursive calculation. Also, notice the default option assigns \( t^? \), in order to cover any unexpected case of a relation that the system might not identify.

To illustrate the recursive process, consider (55), which is fully annotated, with the calculation of CT and PT elaborated in a sequence of steps, starting from the main predicate:

(55) *Mary was glad* \(+/+\), *nu+, ct+, pt+* *that she did not fail* \(-/-\), *nu−, ct−, pt−* *to avoid* \( ?/?, nu+, ct−, pt−\) *meeting* \( ?/?, nu+, ct−, pt−\) *John*

a. sentence main predicate \( \rightarrow \) glad\(^{ct+}\)

b. glad\(^{nu+, ct+}\) \( \rightarrow \) glad\(^{pt+}\)

c. glad\(^{+/+, pt+}\) \( \rightarrow \) fail\(^{ct+}\)

d. fail\(^{nu−, ct+}\) \( \rightarrow \) fail\(^{pt−}\)

e. fail\(^{−/+, pt−}\) \( \rightarrow \) avoid\(^{ct+}\)

f. avoid\(^{nu+, ct+}\) \( \rightarrow \) avoid\(^{pt+}\)

g. avoid\(^{−/?+, pt+}\) \( \rightarrow \) meet\(^{ct−}\)

h. meet\(^{nu+, ct−}\) \( \rightarrow \) meet\(^{pt−}\)

A crucial and delicate feature in the recursion step is detecting predicate-complement relations. It is also imperative to distinguish, as accurately as possible, between finite and infinitival complements, because the signatures of many predicates change between the two (see discussion in Subsection 7.4.1). Basically, my implementation assumes that a predicate parsed either as a “clausal complement with external subject” (XCOMP relation), or as an “infinitival modifier” (INFMOD relation), is an infinitival (or gerundive) complement. All other complement relations are considered finite complements. Notice that, according to the cited theoretical and empirical work on natural logic, subjects do not take part in the recursive process, i.e., factives and implicatives do not affect them as they do objects.

Other implementation details regarding language diversity include accommodating for coordinated complements, various complementizers, passivation and apposition. For instance, the argument of a *there-is* expletive
construction is parsed as a subject (nsubj or csubj), although my system will make an exception, and treat it as an object (dobj or ccomp), so that it participates in the recursive process. See, for example:

(56) There is something\textsubscript{nsubj} rotten in the kingdom of Denmark ⇒ There is something\textsubscript{dobj} rotten in the kingdom of Denmark

These and other implementation details are documented in the download package.

**CT presupposition annotation rules - in tandem with the recursive process** The following rules are like the previous batch of annotation rules concerned with CT presupposition, except that these also depend the precondition that the governing node of the LHS have a specific CT value (ct+), set by one of the rules in R3.1:

**R3.2:** Generic presupposition - the following constructions presuppose their main predicate, annotating it with ct+

(a) Relative clause modifying a ct+ (definite) noun phrase

(57) I want that car\textsuperscript{ct+} that never breaks down ⇒ I want that car\textsuperscript{ct+} that never breaks\textsuperscript{ct+} down

(58) I want a car\textsuperscript{ct?} that never breaks down ⇒ (no annotation)

(b) Participle clause (passive voice relative clauses) modifying a ct+ node

(59) Gila stood\textsuperscript{ct+} and watched for hours, tucked away in the closet ⇒ Gila stood\textsuperscript{ct+} and watched for hours, tucked\textsuperscript{ct+} away in the closet

Figure 17 illustrates my schema for the presupposition of participle modifiers.

While the rules in R3.1 set the initial CT annotations necessary for the recursive CT calculation, these rules may also be fed by it. Consider this example:
(60) \textit{Gal managed } +/-\textit{ct+ a building ct+, which Ginger failed }-/+\textit{ct+ to sell }ct-\textit{.}

Here, in order to correctly annotate ct- on \textit{sell}, it is necessary to apply a relative clause annotation rule and make recursive entailments intermittently, as follows:

1. Recursive implication of \textit{managed }+/-.nu+.ct+: \textit{building ct-}\rightarrow ct+ 
2. Relative clause annotation rule, matched against \{\textit{building ct+, that, failed}\}: \textit{failed ct-}\rightarrow ct+ 
3. Recursive implication of \textit{failed }-/+.nu+.ct+: \textit{sell ct-}\rightarrow ct-

This kind of example points out that the relative clause annotation rules (and potentially others as well) are interdependent with the recursive CT calculation, and that these two elements should be applied in tandem, to improve accuracy. For this reason, I apply R3.2 to the entire tree once each recursion step, i.e., each time a new node is visited, vis a vis the recursive formula presented above.

7.6 Evaluation and Analysis

Although the integration of TruthTeller into BiuTee is still preliminary, as described in the previous subsection, and though substantial work is warranted to explore its full potential, this subsection presents a few initial evaluations of its quality and its contribution to the TE task. Firstly, I estimate the accuracy of TruthTeller’s annotations, i.e., how likely is it to choose the correct annotation values for each predicate, compared to a manually constructed gold standard of 50 annotated sentences from RTE datasets. Secondly, I estimate the recall and precision of each CT and PT value \{ct+, ct-, ct?, pt+, pt-, pt?\} in a similar way. Thirdly, I assess the global impact on the performance of a working RTE system, BiuTee, using ablation tests.

7.6.1 Manual Accuracy Evaluation

In this subsection I report a manual accuracy evaluation of TruthTeller, assessing how often it chooses the correct annotation value for each predicate, as determined in the gold standard annotations, which I manually constructed by analysing sentences from RTE datasets.

I randomly sampled 50 sentences, 25 from the RTE5 Test dataset, and 25 from RTE6 Test. I parsed them (with EasyFirst) and manually annotated every predicate in them, based on the definitions in Subsection 7.4. In total, there were 153 predicates. I then ran TruthTeller and compared its annotations with the manual ones. As a baseline, I computed the matching
Table 15: The accuracy measures for TruthTeller’s 4 annotations, based on an analysis of 50 random RTE sentences, containing 153 predicates in total. The right column gives the accuracy for the baseline, which is the naive annotator that always annotates the most common value of each annotation type: \{+/?, NU+, CT+, PT+\}.

<table>
<thead>
<tr>
<th>Annotation Type</th>
<th>TruthTeller Accuracy</th>
<th>Baseline Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature</td>
<td>89.5%</td>
<td>81%</td>
</tr>
<tr>
<td>NU</td>
<td>98%</td>
<td>97.3%</td>
</tr>
<tr>
<td>CT</td>
<td>90.8%</td>
<td>78.4%</td>
</tr>
<tr>
<td>PT</td>
<td>89%</td>
<td>77%</td>
</tr>
</tbody>
</table>

Table 16: Signature annotation errors committed by TruthTeller in the manual evaluation.

<table>
<thead>
<tr>
<th>Signature Error Type</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet expansion</td>
<td>9</td>
</tr>
<tr>
<td>Unclassified predicate</td>
<td>6</td>
</tr>
<tr>
<td>Manual misclassification</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>16</td>
</tr>
</tbody>
</table>

Signature Errors There were 16 signature errors, whose breakdown is given in Table 16 and examples for the main causes of each error type follow. Out of the 16 times TruthTeller gave a predicate a wrong signature, 9 involved incorrect lexicon entries, created in the process of automatically expanding the manually-constructed kernel with WordNet synonyms and entailments (see Subsection 7.5.1). For example, in one instance the verb *allege* was annotated as a +/? implicative, while it actually behaves like a
Table 17: Confusion matrix between the CT annotation by TruthTeller and the gold standard.

<table>
<thead>
<tr>
<th>TruthTeller</th>
<th>Gold Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>108 0 2</td>
</tr>
<tr>
<td>-</td>
<td>0 0 0</td>
</tr>
<tr>
<td>?</td>
<td>12 0 31</td>
</tr>
</tbody>
</table>

Table 17: Confusion matrix between the CT annotation by TruthTeller and the gold standard.

? regular. This probably happened because it is a Wordnet-synonym of report, which I manually listed as a +?.

6 more signature annotation errors involved predicates that were not listed in my (single word) signature lexicon (neither the manual nor the expanded version), thus getting the default +? signature, while their real signature is different. These last two classes of errors may suggest that there is much room for improving the coverage of my single-word lexicon. Finally, I found 1 case in which a predicate was misclassified in the manually-constructed kernel of the lexicon.

NU Errors Out of a total of 3 NU errors, 2 were related to caveats in the NU annotation rules. Consider:

(a) The show nearly lasted a month

(b) The show has been produced on television and seen nearly every weekday...

Normally, nearly has a negating effect on the predicate, like in (a). However, in (b) the same adverb does not negate the predicate, nor any of its arguments, since it is embedded in an adjunct temporal phrase, and my NU annotation rules fail to make this distinction. Significantly, in most cases the parser also fails to tag such temporal phrases with the standard tmod relation, thus making it hard to formalize rules that do make this distinction.

The last NU error was caused by a parser error, similar to the 11 parser errors that caused CT errors, discussed below.

CT Errors TruthTeller annotated 14 predicates with a wrong CT value, and below I discuss the nature of these errors. Since CT and PT are the system’s main output, I enrich my perspective on the two annotations with their respective confusion matrices, comparing each TruthTeller annotation with the gold standard.

The confusion matrix for CT is given in Table 17. First of all, the concentration of CT annotations on the main diagonal reaffirms the high level of accuracy of the CT annotations, noted above. Furthermore, according to the matrix, the recall for CT+ is 90% and the precision is 98.2%, while
The recall for \( \text{ct}^- \) is 93.9% and the precision is 72.1%. There were no instances of \( \text{ct}^- \) negation in the sampled sentences, which makes it impossible to calculate its recall and precision. Because \( \text{ct}^- \) seems to be so rare in the dataset, further below I describe other experiments which specifically inspect the recall and precision of TruthTeller on \( \text{ct}^- \).

The breakdown of the 14 CT errors is given in Table 18. It shows that parser errors were also the predominate cause for CT errors. To demonstrate this type of error, consider the sentence:

They (the advisers) said I \text{are in Iraq} because they \text{believed} Saddam Hussein \text{had} weapons of mass destruction, that the world’s a better place with Saddam gone and that we’re making the world a safer place with what we’re doing over there, Sheehan said in a telephone interview after the meeting.

In this case, the parser analysed the verb \text{had} as a modifier under \text{are in Iraq}, instead of under \text{believed}. This inevitably caused TruthTeller to annotate \text{had} with \text{ct+} instead of \text{ct?}.

The system also exhibited 2 cases of misinterpreting syntactic relations, causing it to mix up finite complements with non finite, with adjuncts (non-complements). Recall that in Subsection 7.4 I mention that signatures are sensitive to complement finiteness, but conceal this fine distinction throughout most of this study. Consider:

The U.S. government is \text{considering}^{+/+}({\text{finite}}) \text{whether to make},_{\text{comp}} \text{morning-after birth control available}...

In this sentence, the matrix predicate \text{considering} is a \(+/+\) factive with regards to finite complements (\text{considering one course of action versus another}), and is a \( ?/? \) regular predicate with regards to non finite complements (\text{considering to act or not to act}). In addition, the parser labelled \text{make} with a \text{ccomp}, “clausal complement”, relation to \text{considering}, which is predomi-nately used for finite complements, and in rare cases such as this, for non finite complements. Since TruthTeller relies mainly on the parsed relation in order to interpret the type of complement at hand, it treated \text{make} as a finite (presupposed) complement, instead of non finite.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parser</td>
<td>11</td>
</tr>
<tr>
<td>Complement identification</td>
<td>2</td>
</tr>
<tr>
<td>Signature</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>14</td>
</tr>
</tbody>
</table>

Table 18: CT annotation errors committed by TruthTeller in the manual evaluation.
Table 19: Confusion matrix between the PT annotation by TruthTeller and the gold standard.

<table>
<thead>
<tr>
<th>TruthTeller\Gold Standard</th>
<th>+</th>
<th>-</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>103</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>?</td>
<td>14</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>

The last CT error was a direct consequence of another error in the governing predicate’s signature annotation (discussed above), on which the recursive CT calculation depends.

**PT Errors** The confusion matrix for PT is given in Table 19. Like the CT confusion matrix, it reaffirms the high accuracy measured above, where the derived recall for $pt^+$ is 87.3% and the precision is 98%, while the recall for $pt^?$ is 91.4% and the precision is 70%. Also like in the CT confusion matrix, I can see that there were scarce examples of $pt^-$ negation in the sampled dataset, which led me to focus the following two experiments on $pt^-$ recall and precision, as well as on $ct^-$.

Since PT is calculated as the binary product of NU and CT, each PT error depends on the presence of other errors in the NU and/or CT of the same predicate. Therefore, the causes for PT errors are simply combinations of the NU and CT errors, discussed above.

7.6.2 Precision and Recall of Negative Annotations

In light of the scarcity of negative CT and PT annotations found in the accuracy evaluations of Subsection 7.6.1, in this subsection I focus on the precision and recall of TruthTeller’s $ct^-$ and $pt^-$ annotations. First, I report a precision estimation, based on the correctness of a given set of system annotations on a random sample of RTE6 Test sentences. Second I report a recall estimation, based on how many negative classifications my system correctly labelled, out of a set of over 50 $ct^-$ and $pt^-$ annotations from RTE6 Test, obtained from an independent human gold standard.

**Precision of Negative Annotations** In this evaluation, I used my system to annotate the RTE6 Test corpus, and randomly selected a set of annotated sentences, such that contained over 50 $ct$-s and 50 $pt$-s. The set consisted of 96 sentences, including 50 predicates annotated with $ct^-$ and 124 with $pt^-$. I found that $pt^-$ is generally much more frequent because it is triggered in almost all cases of $ct^-$, as well as by a host of other constructions (involving $nu^-$ negation).
Table 20: CT- and PT- precision estimation, based on a random sample of sentences from the RTE6 Test dataset, annotated by TruthTeller. Each occurrence is a predicate annotated with a negative value.

<table>
<thead>
<tr>
<th>Value</th>
<th>Occurrences</th>
<th># Errors</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>ct-</td>
<td>50</td>
<td>11</td>
<td>78%</td>
</tr>
<tr>
<td>pt-</td>
<td>124</td>
<td>21</td>
<td>83%</td>
</tr>
</tbody>
</table>

Table 21: Breakdown of the 11 erroneous CT- annotations, found in a random sample of annotated RTE6 Test sentences.

The results of the evaluations are given in Table 20. It shows rather high precision on both CT- annotations (78%) and PT- (83%). Table 21 holds the breakdown of the 11 erroneous CT- annotations. All its error types have already been mentioned and discussed around Table 16 and Table 18, except for the annotation rule error. This one can be partly attributed to a parse error and partly to an “over-fitted” rule, regarding the sentence:

*Because the SUV was “lodged\textsuperscript{ct+}\textsuperscript{pt+} on the tracks,” it was unable to move, he said.*

The sentence features a presupposed adverbial clause *Because the ... tracks*, which should get both CT+ and PT+. However, the parser failed to tag the syntactic relation between the adverbial clause and its dominating predicate *unable* with an indicative label such as advcl, and used the generic dep relation instead. Thus, my adverbial clause rule (see Subsection 7.5.3) failed to match against the clause, and it was annotated erroneously with CT- and PT-. Still, I attribute this error to my annotation rule, because it brought to my attention that the rule would actually perform better if it didn’t require to match against any specific relation to the parent clause at all, but rather it only identified the opening word of the clause in my list of presupposing adverbials, i.e., *because, since, after etc.*

Table 22 holds the breakdown of the 21 erroneous PT- annotations. All its error types have already been mentioned and discussed above.
Table 22: Breakdown of the 21 erroneous pt- annotations, found in a random sample of annotated RTE6 Test sentences.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parser</td>
<td>7</td>
</tr>
<tr>
<td>Annotation rule</td>
<td>4</td>
</tr>
<tr>
<td>Signature (manual)</td>
<td>3</td>
</tr>
<tr>
<td>Signature (WordNet expansion)</td>
<td>3</td>
</tr>
<tr>
<td>Signature (unclassified predicate)</td>
<td>2</td>
</tr>
<tr>
<td>Complement identification</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 23: ct- and pt- recall estimation, based on a random sample of sentences from the RTE6 Test dataset, annotated by a human expert. Each occurrence is a predicate annotated with a negative value.

Recall of Negative Annotations  In this evaluation, I asked an independent expert reader to identify cases of ct- and pt- in the RTE6 Test dataset, according to my definitions for annotation types, given in Subsection 7.4. Each sentence in the dataset was read individually and in random order.

She read through the entire dataset, comprised of a total of 3333 sentences, and identified only 40 predicates with ct-. This figure suggests that negative CT is very rare in RTE texts. Also, out of the first 350 sentences read, 50 instances of pt- were identified, suggesting that negative PT is also rare, but still significantly more abundant than negative CT. Note that both these figures are congruent with the scarcity of negative annotations noted in Subsection 7.6.1. According to this analysis, the frequency of ct- in the corpus can be estimated at 1.2%, and of pt- at 14.3%. I then continued to annotate the same sentences with TRUTHTELLER, and derived its recall relative to my gold standard of negative annotations, displayed in Table 23.

I can see there that TRUTHTELLER has a rather low recall estimation for ct- cases (47.5%), while the recall estimation for pt- cases is rather high (74%). One possible explanation for this contrast is that ct- occurs only in rare constructions, in which the recursive formula in Subsection 7.5.3 yields a negative value, while pt- occurs also in relatively common constructions, such as most sentences with a simple negation word, e.g., Susan never forgot pt- her true passion.

The breakdown of the ct- recall errors is given in Table 24 and below.
<table>
<thead>
<tr>
<th>Error Type</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature (unclassified phrasal verb)</td>
<td>5</td>
</tr>
<tr>
<td>Annotation rule</td>
<td>5</td>
</tr>
<tr>
<td>Parser</td>
<td>4</td>
</tr>
<tr>
<td>Signature ambiguity</td>
<td>4</td>
</tr>
<tr>
<td>Signature (unclassified single word)</td>
<td>2</td>
</tr>
<tr>
<td>Signature (manual misclassification)</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 24: Breakdown of the 21 \textit{ct}- recall errors, measured against a human gold standard annotation of \textit{ct} in the entire RTE6 Test corpus.

I demonstrate each category.

Firstly, 5 errors were caused by not identifying the signatures of certain phrasal verbs (combinations of a verb and a noun). For instance, the phrase \textit{deny request} to \textit{X}, has a \textit{-/?} signature with regards to \textit{X}, which is different than the (default) \textit{+/?} signature that \textit{request} has. This information is imperative in the following sentence, for correctly annotating the argument of \textit{request}, which is \textit{combine}:

\textit{Manhattan judge denies request$^{-/?}+/?$ to combine$^{t-}\{ct\}$ Kozlowski larceny, tax}

Nonetheless, since currently the only lexicon for identifying and classifying such verb phrases is [Karttunen 2012], and it doesn’t cover this phrase, the \textit{+/?} signature is used, which causes the argument \textit{combine} to get \textit{ct+} instead of \textit{ct-}.

5 more recall errors were found in relatively uncommon constructions that were not covered by my algorithm, as implemented by the annotation rules. For example, consider:

\textit{At times, he said, he lacked the 10 cents to print out even one page of his briefs and did not have the money to mail his appeal...}

Usually, the definite article \textit{the} implies that its noun is presupposed, and thus gets a \textit{ct+} (see rule R3.1/c/i in Subsection 7.5.3). Here, however, the expressions \textit{lacked the X to Y} and \textit{did not have the X to Y} present two exceptional constructions, where the noun in negatively entailed, and should have a \textit{ct-}. TruthTeller lacked a rule for recognizing these exceptional cases (because they are rather rare, and require a lexicon of relevant predicates), and therefore the two \textit{ct-} annotations in this sentence were missed.

The 4 parser errors, along with the “unclassified single word” and “manual misclassification” errors were already discussed in their corresponding error categories in Subsection 7.6.1 and need no further elaboration here. The last 4 errors are attributed to predicates with ambiguous signatures,
Table 25: Breakdown of the 13 pt- recall errors, measured against a human gold standard annotation of pt- in about 350 random sentences from the RTE6 Test corpus.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parser</td>
<td>7</td>
</tr>
<tr>
<td>Annotation rule</td>
<td>5</td>
</tr>
<tr>
<td>Signature ambiguity</td>
<td>1</td>
</tr>
</tbody>
</table>

such as reject X, which has two distinct signatures for two of its senses: rejecting an action vs. rejecting a proposal. Compare:

(1) The FDA rejected?/pt+ nonprescription sales?ct− of emergency contraception…

(2) The FDA rejected+/pt+ a proposal?kt+ to sell nonprescription emergency contraception…

The first sentence demonstrates that rejecting an action entails that the action did not take place, while the second shows that rejecting a proposal entails the existence/action of that proposal. Currently, TRUTHTELLER lacks the sophistication to distinguish between these two senses. Therefore, when annotating the first example, the (default) +/? signature was used, yielding a ct+ on the direct complement sales, instead of the correct value ct−.

The breakdown of the 13 pt- errors is given in Table 25. All the error types have already been discussed above.

7.6.3 Impact on System Results

In this subsection I report two ablation tests that assess TRUTHTELLER’s global impact on the performance of a working RTE system, BiuTee. Table 26 shows the global F1, recall and precision measures of the system in its baseline configuration, (with WordNet and Bap lexical resources), like in Section 5, and with the addition of TRUTHTELLER, running as a prepro- cess step. It was run on the RTE5 and RTE6 datasets. As before, I use the standard definitions of precision, recall and F1 typically used in NLP benchmarks.

I see that straightforwardly using the resource increases the F1 measure on RTE6 by almost 1%, from 48.75% in the baseline run, to 49.64% in the rule-base test run. On RTE5, the improvement is barely significant, by about 0.2%. The above results are statistically significant ($p < 0.02$ using Mcneamar test).

12The baseline results differ slightly from those in Table 11 because of version changes in BiuTee itself.
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Recall</th>
<th>Precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline RTE6</td>
<td>43.39%</td>
<td>55.63%</td>
<td>48.75%</td>
</tr>
<tr>
<td>TruthTeller Test RTE6</td>
<td>43.81%</td>
<td>57.26%</td>
<td>49.64%</td>
</tr>
<tr>
<td>Baseline RTE5</td>
<td>79.76%</td>
<td>64.5%</td>
<td>71.32%</td>
</tr>
<tr>
<td>TruthTeller Test RTE5</td>
<td>80%</td>
<td>64.7%</td>
<td>71.54%</td>
</tr>
</tbody>
</table>

Table 26: The F1, recall and precision measures of the system, run on the RTE6 and RTE5 test suites, in baseline configuration (with WordNet and Bap lexical resources), and with TruthTeller.

7.7 Conclusions

In this section I have presented TruthTeller, a novel algorithm and system that takes the syntactic parse tree of a given sentence, and identifies the semantic truth value of each predicate and clause. In contrast with previous work, it is the first such system meant to serve the natural language inference research community as an open source tool, enriching conventional syntax tree representation. Some possible uses for these annotations are: inferring parts of a sentence from the whole, improving similarity (and contradiction) measures between texts, and improving the accuracy of entailment rule matching. As a side product, TruthTeller also annotates predicate implication signature and negation. The download package also includes a lexicon of predicates and their signatures, the largest of its kind and the first to be made publicly available. Both the lexicon and TruthTeller’s annotations are shown to have good accuracy, recall and precision.

The process of constructing TruthTeller, and its evaluation, suggest several straightforward directions for future work. First and foremost, it is important to integrate it within a concrete entailment engine, and to investigate how to optimally make use of the annotations in the TE task (probably focusing on PT). Afterwards, I would like to model and incorporate monotonicity into the system, similar in spirit to (MacCartney and Manning 2007). Also, Subsection 7.6.2 demonstrates that identifying CT- and PT-re mains a challenge, which I plan to address. Those and other results indicate that augmenting and refining the single word signature lexicon will probably yield a noticeable improvement in performance.

8 Acknowledgements

A special thanks goes to my two advisors, Prof. Ido Dagan and Dr. Roni Katzir, for showing me the ropes, and for seeing me through this long laborious and thrilling journey.

Thanks also to Prof. Fred Landman of the Linguistics Department at Tel Aviv University for his help in forming my semantics of presupposition.
A Formal Definition and Application of Entailment Rules

This subsection defines entailment rules, and the formalism for applying them on parse trees, which I assume that entailment systems follow.

A rule ‘$L \Rightarrow R$’ is composed of two templates, $L$ on the left-hand-side (LHS) and $R$ on the right-hand-side (RHS). Templates are dependency parse subtrees which may contain variable nodes. These are regular nodes that have no specified lemma, so that they match any lemma. Figure 1(a) shows active-to-passive transformation rule, and Figure 1(b) illustrates its application. Technically, there are two distinct types of rules: substitution and extraction, described ahead.

The rule application algorithm is given in Figure 18. One rule application, involving one match between the rule and a source tree $s$, generates one derived tree. Likewise, consecutive applications of one rule generate a set $D$ of derived trees (consequents) from $s$, through the following steps:

```
Input: a source tree $s$, a rule $R : L \Rightarrow R$
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 $u \in l$ and $u \in r$ do
      for each daughter $m$ of $u$ such that $m \in l$ do
        Copy the subtree of $s$ rooted in $m$ under $u$ 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 // extraction rule
    $d \leftarrow r$
add $d$ to $D$
```

Figure 18: The algorithm for applying a rule to a tree. Taken from Bar-Haim et al. (2009).

and implicative entailments.
1. \textit{L matching} First, matches of \textit{L} in the source tree \textit{s} are sought. \textit{L} is matched in \textit{s} if there exists a one-to-one node mapping function \textit{f} from \textit{L} to \textit{s}, such that:

(a) For each node \textit{u} in \textit{L}, \textit{f(u)} matches the lemma and POS of \textit{u}. Variables match any lemma value in \textit{f(u)}.

(b) For each edge \textit{u} ⇒ \textit{v} in \textit{L}, there is an edge \textit{f(u)} ⇒ \textit{f(v)} in \textit{s}, with the same dependency relation.

If matching fails, the rule is not applicable to \textit{s}. In my example in Figure 1(b), the variable \textit{V} is matched in the verb \textit{see}, \textit{N1} is matched in \textit{Mary} and \textit{N2} is matched in \textit{John}. If matching succeeds, then the following is performed for each match found.

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

3. \textit{Derived tree generation} Let \textit{r} be the instantiated \textit{R}, along with its descendants, copied from \textit{L} through alignment, and \textit{l} be the subtree matched by \textit{L}. The formalism has two methods for generating the derived tree \textit{d}: substitution and extraction, as specified by the rule type. Substitution rules specify modification of a subtree of \textit{s}, leaving the rest of \textit{s} unchanged. Thus, \textit{d} is formed by copying \textit{s} while replacing \textit{l} (and the descendants of \textit{l}'s nodes) with \textit{r}. This is the case for the passive rule, as well as for lexical rules such as \textit{buy} ⇒ \textit{purchase}. By contrast, extraction rules are used to make inferences from a subtree of \textit{s}, while the other parts of \textit{s} are ignored and do not affect \textit{d}. A typical example is inferring a proposition embedded as a temporal clause in \textit{s}. In this case, the derived tree \textit{d} is simply taken to be \textit{r}. Figure 2 presents such a rule, which enables to derive propositions that are embedded within temporal modifiers. Note that the derived tree does not depend on the main clause. Applying this rule to the bottom tree in Figure 1(b) yields the proposition \textit{John saw beautiful Mary yesterday}. 
B Mapping from Penn Tree Bank POS Set to the Reduced POS Set

Table 27 defines the conversion from the Penn Tree Bank POS set, as specified in the Stanford dependency relations standard, which I adopt, to my proprietary reduced POS set.

C Description of BiuTee, the Bar Ilan University Textual Entailment Engine

At the preprocessing stage, BiuTee parses all the texts and hypotheses using EasyFirst parser (Goldberg and Elhadad 2010), and adds coreference information using ArkRef (Haghighi and Klein 2009). Then, using machine learning, it finds the simplest and most reliable proofs to T - H pairs in the space of all possible proofs. The transformations are loaded from inference rule-base resources, such as my syntax-based resource, WordNet (Fellbaum 1998, Miller 1995), Wikipedia (Shnarch et al. 2009) and many others. In many cases, the resources are insufficient for the task, and therefore the system also uses a built-in set of “on the fly” transformations. These can manipulate a tree in any desirable way (adding, deleting, altering nodes and edges etc.), but have no linguistic justification. Hence the system tries to apply them as little as possible.

References


Table 27: The conversion from the Penn Tree Bank POS set, as specified in the Stanford dependency relations standard, to my proprietary reduced POS set

<table>
<thead>
<tr>
<th>Reduced POS</th>
<th>Penn POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>VERB</td>
</tr>
<tr>
<td>VBD</td>
<td>VERB</td>
</tr>
<tr>
<td>VBG</td>
<td>VERB</td>
</tr>
<tr>
<td>VBN</td>
<td>VERB</td>
</tr>
<tr>
<td>VBP</td>
<td>VERB</td>
</tr>
<tr>
<td>VBZ</td>
<td>VERB</td>
</tr>
<tr>
<td>NN</td>
<td>NOUN</td>
</tr>
<tr>
<td>NNS</td>
<td>NOUN</td>
</tr>
<tr>
<td>NNP</td>
<td>NOUN</td>
</tr>
<tr>
<td>NNPS</td>
<td>NOUN</td>
</tr>
<tr>
<td>JJ</td>
<td>ADJECTIVE</td>
</tr>
<tr>
<td>JJR</td>
<td>ADJECTIVE</td>
</tr>
<tr>
<td>JJS</td>
<td>ADJECTIVE</td>
</tr>
<tr>
<td>DT</td>
<td>DETERMINER</td>
</tr>
<tr>
<td>WDT</td>
<td>DETERMINER</td>
</tr>
<tr>
<td>PDT</td>
<td>DETERMINER</td>
</tr>
<tr>
<td>PRP</td>
<td>PRONOUN</td>
</tr>
<tr>
<td>PRP$</td>
<td>PRONOUN</td>
</tr>
<tr>
<td>WP</td>
<td>PRONOUN</td>
</tr>
<tr>
<td>WP$</td>
<td>PRONOUN</td>
</tr>
<tr>
<td>RB</td>
<td>ADVERB</td>
</tr>
<tr>
<td>RBR</td>
<td>ADVERB</td>
</tr>
<tr>
<td>RBS</td>
<td>ADVERB</td>
</tr>
<tr>
<td>WRB</td>
<td>ADVERB</td>
</tr>
<tr>
<td>CC</td>
<td>PREPOSITION</td>
</tr>
<tr>
<td>IN</td>
<td>PREPOSITION</td>
</tr>
<tr>
<td>RP</td>
<td>PREPOSITION</td>
</tr>
<tr>
<td>TO</td>
<td>PREPOSITION</td>
</tr>
<tr>
<td>RCB</td>
<td>PUNCTUATION</td>
</tr>
<tr>
<td>LCB</td>
<td>PUNCTUATION</td>
</tr>
<tr>
<td>LRB</td>
<td>PUNCTUATION</td>
</tr>
<tr>
<td>RRB</td>
<td>PUNCTUATION</td>
</tr>
<tr>
<td>LS</td>
<td>PUNCTUATION</td>
</tr>
<tr>
<td>SYM</td>
<td>OTHER</td>
</tr>
<tr>
<td>CD</td>
<td>OTHER</td>
</tr>
<tr>
<td>EX</td>
<td>OTHER</td>
</tr>
<tr>
<td>FW</td>
<td>OTHER</td>
</tr>
<tr>
<td>MD</td>
<td>OTHER</td>
</tr>
<tr>
<td>POS</td>
<td>OTHER</td>
</tr>
<tr>
<td>UH</td>
<td>OTHER</td>
</tr>
</tbody>
</table>


Karttunen L (2012) Simple and phrasal implicatives. In: To be published


85