Natural Language Engineering **21** (1): 3–47. © Cambridge University Press 2013
doi:10.1017/S1351324913000156

Unsupervised acquisition of entailment relations from the Web

IDAN SZPEKTOR[1], HRISTO TANEV[2], IDO DAGAN[3], BONAVENTURA COPPOLA[4] and MILEN KOUYLEKOV[5]

[1]*Yahoo! Research, Haifa, Israel*
e-mail: idan@yahoo-inc.com
[2]*JRC, Ispra, Italy*
e-mail: htanev@gmail.com
[3]*Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel*
e-mail: dagan@cs.biu.ac.il
[4]*IBM Thomas J. Watson Research Center, Yorktown Heights, NY*
e-mail: bcoppola@us.ibm.com
[5]*CELI s.r.l., Torino, Italy*
e-mail: kouylekov@celi.it

(*Received 29 March 2011; revised 23 June 2013; accepted 26 June 2013; first published online 30 July 2013*)

Abstract

Entailment recognition is a primary generic task in natural language inference, whose focus is to detect whether the meaning of one expression can be inferred from the meaning of the other. Accordingly, many NLP applications would benefit from high coverage knowledgebases of paraphrases and entailment rules. To this end, learning such knowledgebases from the Web is especially appealing due to its huge size as well as its highly heterogeneous content, allowing for a more scalable rule extraction of various domains. However, the scalability of state-of-the-art entailment rule acquisition approaches from the Web is still limited. We present a fully unsupervised learning algorithm for Web-based extraction of entailment relations. We focus on increased scalability and generality with respect to prior work, with the potential of a large-scale Web-based knowledgebase. Our algorithm takes as its input a lexical–syntactic template and searches the Web for syntactic templates that participate in an entailment relation with the input template. Experiments show promising results, achieving performance similar to a state-of-the-art unsupervised algorithm, operating over an offline corpus, but with the benefit of learning rules for different domains with no additional effort.

1 Introduction

A fundamental phenomenon in natural language is the surface-level variability of semantic expressions – having different ways to convey the same meaning. In many natural language processing (NLP) applications, such as Question Answering, Information Extraction (IE), Information Retrieval, Multi-Document Summarization and Machine Translation (Jacquemin 1999; Moldovan and Rus 2001; Hermjakob, Echihabi and Marcu 2003; Harabagiu and Hickl 2006; Lloret *et al.* 2008; Mirkin, Dagan and Shnarch 2009), the need to identify different sentences that express the same meaning have been recognized as critical for reaching high performance. For example, given the question 'Who killed Kennedy?', a

Question Answering system would need to identify that the sentence '*Oswald assassinated Kennedy*' contains the correct answer; while a Document Summarization system would like to identify that the sentences '*Yahoo acquired Overture*' and '*Overture was bought by Yahoo*' describe the same event in order to include only one of them in a summary. As a result, there is an increasing interest in modeling surface-level variability (Moldovan and Rus 2001; Monz and de Rijke 2001; Condoravdi *et al.*2003; Durme *et al.* 2003). One specific approach, Textual Entailment (Dagan and Glickman 2004), has drawn a lot of attention in recent years as a generic framework for applied semantic inference (Bar-Haim *et al.* 2006; Dagan, Glickman and Magnini 2006; Giampiccolo *et al.* 2007; Giampiccolo *et al.* 2008; Bentivogli *et al.* 2009; Bentivogli *et al.* 2010). We follow the notions of this paradigm for semantic inference in this work.

A major component in the modeling of surface-level variability consists of paraphrase rules, pairs of language expressions that can replace each other in a sentence without changing its meaning. Indeed, the above example about Yahoo! demonstrates a paraphrase. Paraphrase rules range from synonyms, such as 'purchase $\Leftrightarrow$ buy', to complex expressions, such as 'kick the bucket $\Leftrightarrow$ die' and '*X* manufactures *Y* $\Leftrightarrow$ *Y* is a product made by *X*'. In this work we focus on paraphrase rules between two templates, text fragments with variables, such as the last example. There are numerous paraphrases in a language and it would be a laborious task to collect them all manually. This perception led to a substantial effort of automatic discovery of paraphrase rules (Barzilay and McKeown 2001; Lin and Pantel 2001; Shinyama *et al.* 2002; Ravichandran and Hovy 2002; Barzilay and Lee 2003; Sekine 2005; Bannard and Callison-burch 2005; Zhao *et al.* 2008).

Following the Textual Entailment paradigm, a more general notion needed for semantic inference is that of entailment rules (Dagan and Glickman 2004). An entailment rule is a *directional* relation between two language expressions, where the meaning of one can be entailed (inferred) from the meaning of the other. For example, the sentence '*Yahoo acquired Overture*' entails the meaning of the sentence '*Yahoo owns Overture*', but not vice versa. This semantic relation can be described by the entailment rule '*X* acquire *Y* $\Rightarrow$ *X* own *Y*'. Entailment rules provide a broad framework for representing and recognizing surface-level variability and are a generalization of paraphrases, which correspond to bidirectional entailment rules (*e.g.* '*X* purchase *Y* $\Leftrightarrow$ *X* buy *Y*').

Most methods for identifying Textual Entailment do not represent meanings explicitly. Instead, entailment inferences are performed directly over lexical–syntactic expressions.[1] Hence, NLP applications that require entailment inferences would largely benefit from broad coverage knowledgebases of entailment rules (Bar-Haim, Szpektor and Glickman 2005; Bar-Haim *et al.* 2006). When discussing broad coverage resources, we refer to two complementing targets. The first is that we would like such a knowledgebase to provide from a handful to dozens of entailment rules for a large number of templates. Focusing on verbs, such a rule-set may sum up to hundreds of thousands and even millions of rules, given that a typical lexicon includes tens of thousands of words. As a second target, we also expect from a broad coverage open domain resource to provide rules for each of the

[1] See summaries of these studies in Bar-Haim *et al.* (2006), Dagan *et al.* (2006), Giampiccolo *et al.* (2007, 2008) and Bentivogli *et al.* (2009, 2010).

possible meanings of a template. For example, we expect to find for '*X* acquire *Y*' both '*X* acquire *Y* ⇔ *X* buy *Y*' and '*X* acquire *Y* ⇔ *X* learn *Y*'.

Aiming at a broad coverage knowledgebase construction, in this work we are interested in exploiting the Web for automatically learning pairs of language expressions that participate in an entailment relation. The advantages of using the Web lie both in its sheer size – it is the largest available textual resource, as well as in its highly heterogeneous content, which should help in finding rules for the different meanings of target templates. The prominent approach to Web-based acquisition of paraphrases involves the exploitation of sets of multiple context words, termed here *anchor-sets*, for the identification of sentences containing paraphrases (Ravichandran and Hovy 2002; Suchanek, Ifrim and Weikum 2006). An anchor-set contains the main participants of an event or a fact. For example, the anchor-set {*X*='*Mendelssohn*', *Y*='*incidental music*'} characterizes the event '*Mendelssohn wrote incidental music*'. Thus, under certain conditions, two sentences can be assumed to convey the same meaning if they contain the same anchor-set. For example, '*Mendelssohn wrote incidental music*' and '*Mendelssohn composed incidental music*', which both contain the above anchor-set, depict the same event. From such pairs of sentences, paraphrases and entailment rules, such as '*X* write *Y* ⇔ *X* compose *Y*', are extracted. The main drawback of such Web-based approaches is that they require that anchor-sets will be provided through manual supervision.

Another problem of entailment relation acquisition in general is the identification of the most appropriate structure of the lexical–syntactic templates that participate in a rule. Prior work that addressed lexical–syntactic templates assumed a fixed template structure in the form of a path between two nouns in the syntactic parse tree of a sentence (Lin and Pantel 2001; Shinyama *et al.* 2002). However, not all templates follow this structure. For example, the syntactic structure of '*X* call *Y* indictable' is not a path.

In this article, we address the two problems introduced above through a combination of two different algorithms. The first is an unsupervised approach for anchor-set acquisition from the Web, overcoming the main scalability limitation of prior Web-based methods. The second algorithm finds the most general repeated template structures that are justified by the data, enabling flexible template structure discovery. Combining these two algorithms, we present TEASE[2], an unsupervised algorithm that extracts numerous anchor-sets automatically and exploits them to learn a large number of entailment relations[3] from the Web. Our algorithm was applied to the acquisition of entailment relations between verbal expressions. It successfully identified several dozen relations on average per each randomly chosen expression. As our experiments show, TEASE is especially suitable to open IE settings, in which entailment rules for various domains should be extracted quickly and without supervision or additional effort, such as domain-specific corpora construction. Examples of entailment relations learned by our algorithm are shown in Table 1 (see additional examples in Table 5 (Section 5) and Table 9 (Section 7.2)).

[2] The TEASE acronym stands for the two main phases in the algorithm: Template Extraction, TE; and Anchor-Set Extraction, ASE.

[3] In this work, we do not learn the entailment direction between the extracted template pairs. Recent work suggest approaches for learning the direction of rules (Bhagat, Pantel and Hovy 2007; Szpektor and Dagan 2008), but this task is out of the scope of the current article.

Table 1. *Examples of correct entailment relations learned by the TEASE algorithm: each learned template participates in an entailment relation with the input verb*

Input template	Learned templates	
X accuse *Y*	*X* sue *Y*	*X* file lawsuit against *Y*
	X blame *Y*	*Y* is named in *X* lawsuit
	X attack *Y*	*X* condemn *Y*
	X 's attack on *Y*	*X* smear *Y*
	X file charges against *Y*	*X* call *Y* indictable
X defeat *Y*	*Y* lose to *X*	*X* destroy *Y*
	X beat *Y*	*X* win *Y*
	X victory over *Y*	*X* conquer *Y*
X demand *Y*	*X* campaign for *Y*	*X* insist on *Y*
	X push for *Y*	*X* add for *Y*
	X in favor for *Y*	*X* call for *Y*
	X call on to make *Y*	*X* urge to impose *Y*
X host *Y*	bring *Y* to *X*	*Y* is played in *X*
	Y is held in *X*	*X* venue of *Y*
	Y come to *X*	*X* play host to *Y*
X preclude *Y*	*X* prevent *Y*	*X* exclude *Y*
	X bar *Y*	*X* deny *Y*
	X deprive *Y*	*X* denial of *Y*
	X oppose *Y*	*X* foreclose *Y*
	X render *Y* inappropriate	*X* suppress *Y*
	Y is inappropriate in light of *X*	*X* is limitation on *Y*
X recover *Y*	*X* seek *Y*	*X* is entitled to *Y*
	X receive *Y*	*X* sue for *Y*
	X claim *Y*	*X* collect *Y*
	X suffer *Y*	pay *Y* to *X*
	X obtain *Y*	*Y* compensate *X*
	X seek recovery of *Y*	*X* pursue claim for *Y*
	give *Y* to *X*	*X* accept *Y*

The rest of the article is outlined as follows. Section 2 reviews the exploitation of paraphrases and entailment rules in applications and describes previous approaches to automatic paraphrase acquisition and to learning template structure. Section 3 presents a formal definition of the task and describes the overall structure of the TEASE algorithm. Sections 4 and 5 describe in detail the two phases of the TEASE algorithm: anchor-set extraction (ASE) and template extraction (TE). Sections 6 and 7 present the evaluation methodology, the experiments we performed, their results, and a comprehensive error analysis. The comparison with DIRT (Lin and Pantel 2001), another state-of-the-art unsupervised large-scale acquisition algorithm, is also discussed. Section 8 concludes and suggests future work.

This work is an extension of the work presented in Szpektor *et al.* (2004), containing a detailed description of the anchor-set extraction step and a significantly more efficient algorithm for the template extraction step (including a complexity analysis). It also includes a new evaluation, comparing TEASE to a manually constructed entailment rule resource from WordNet, as well as additional analysis of our experiments.

2 Background and related work

Inferences about language variability were often handled by practical systems at a 'shallow' semantic level, due to the fact that robust semantic interpretation into logic-based meaning-level representations is not broadly applicable. Until recently, there was no common framework for modeling variability in an application independent manner. Consequently, these inferences were treated mostly independently within individual systems. For example, Lin and Pantel (2001) aimed at improving the coverage of Question Answering systems by learning paraphrase rules, while Shinyama *et al.* (2002) focused on the task of IE.

In recent years, Textual Entailment (Dagan and Glickman 2004) was proposed as a generic framework for recognizing language variability at a shallow semantic level. This approach is based on modeling entailment between language expressions, aiming to identify that the meaning of one expression can be inferred from the meaning of the other. Several studies applied entailment engines to NLP applications, such as Question Answering (Harabagiu and Hickl 2006), IE (Romano *et al.* 2006), Text Summarization (Lloret *et al.* 2008) and Machine Translation (Mirkin *et al.* 2009), showing the benefits of a generic inference engine to these tasks.

One of the prominent knowledge representations in Textual Entailment are *entailment rules* (Bar-Haim *et al.* 2005). An entailment rule is a directional relation $LHS \Rightarrow RHS$, where *LHS* and *RHS* are two *templates*, text fragments with variables, *e.g.* 'X lose to $Y \Rightarrow$ X play against Y'. An entailment rule is considered correct if (at least some) instantiations of the *LHS* variables would entail the meaning of the *RHS* under the same *variable instantiation*, the replacement of the template variables with concrete words or terms. For example, 'X acquire $Y \Leftrightarrow X$ buy Y' is a correct rule (*e.g.* 'Microsoft acquire Skype ⇔ Microsoft buy Skype'), while 'X manufacture $Y \Leftrightarrow X$ buy Y' is incorrect. A pair of templates $\langle T_1, T_2 \rangle$ such that either '$T_1 \Rightarrow T_2$', '$T_2 \Rightarrow T_1$' or '$T_2 \Leftrightarrow T_1$' is defined here as an *entailment relation*, where the entailment direction is not specified. Another type of inference rules that is used frequently in the literature is *paraphrases* (or *paraphrase rules*). There is no precise definition of paraphrases in the literature but they are largely treated as 'expressions that mean the same thing'. In terms of entailment, we view paraphrases as a special case of entailment rules where the two templates entail each other (in both directions), *e.g.* 'X acquire $Y \Leftrightarrow X$ buy Y'.

To complete the description of entailment rules, we note two aspects. First, in our work texts are represented at the syntactic level (we use the Minipar dependency parser; Lin 1998). Thus, a lexical–syntactic template is a connected sub-parse tree with variables at some nodes, *e.g.* '$X \xleftarrow{subj}$ prevent $\xrightarrow{obj} Y$' and '$X \xleftarrow{subj}$ go $\xrightarrow{prep}$ to $\xrightarrow{mod} Y$'. Second, a correct entailment rule should be valid under some contexts, but not necessarily under all contexts (Szpektor, Shnarch and Dagan 2007; Pantel *et al.* 2007; Szpektor *et al.* 2008;

Androutsopoulos and Malakasiotis 2010). The reason is that many predicates are ambiguous while entailment rules typically pertain to a single meaning of the predicate. For example, '*X* acquire *Y* ⇔ *X* buy *Y*' is valid under the instantiation {*X*='*John*', *Y*='*new car*'} but not under the instantiation {*X*='*students*', *Y*='*new language*'}, for which a valid entailment relation would be '*X* acquire *Y* ⇔ *X* learn *Y*'.

For a generic entailment inference system to be effective, a large-scale knowledgebase of entailment rules is needed (Bar-Haim *et al.* 2006). As textual entailment is a relatively new paradigm, prior work mostly concentrated on acquiring paraphrases. We focus our review on the aspect of broad-coverage acquisition in order to present the inherent limitations of previous research, and to show the motivations for our method. As there are no standard benchmarks for paraphrase acquisition, quantitative results of different studies are usually not comparable in a uniform way.

Paraphrase acquisition methods can be described in terms of the methodology they are applying and in terms of the resources they utilize for extracting rules. A convenient way to construct a high-quality rule-set is to extract paraphrase and entailment rules from machine-readable manually constructed lexicons, which also contain information on synonyms and entailment relations between words, such as WordNet and FrameNet (Miller 1995; Baker, Fillmore and Lowe 1998). Entailment rules are harvested from such lexicons mainly by going over the manually specific relations that indicate entailment relations (Mirkin *et al.* 2009; Ben Aharon, Szpektor and Dagan 2010). While this approach generates high-quality rules, its limitations lie in the size and type of rules that can be extracted. First, since annotating entailment relations is not easy, manually constructed lexicons provide only a limited coverage of such relations. Second, in WordNet, which is the most utilized manual resource, most of the annotated relations, such as synonyms and hypernyms, refer to lexical substitutable relations, without arguments, and cannot include rules with syntactic changes, such as '*X* buy from *Y* ⇔ *Y* sell to *X*' (see Section 7.4 for a comparison with our algorithm).

Indeed, to complement rule extraction from manually constructed resources, many studies attempted to extract rules from textual corpora, including bilingual corpora, regular corpora and the Web. These studies include two main approaches for rule learning: (a) *sentence alignment* (Shinyama *et al.* 2002; Ravichandran and Hovy 2002; Ibrahim, Katz and Lin 2003; Sekine 2005; Zhao *et al.* 2008) and (b) *distributional similarity* (Lin and Pantel 2001; Pekar 2006; Szpektor and Dagan 2008). The next subsections review these approaches as they are presented in related work.

2.1 The sentence-alignment approach

The prominent approach for paraphrase learning from corpora is sentence alignment. This approach finds, within sentences, linguistic structures, termed here *templates*, which share the same lexical context elements, termed here *anchors*. For example, the phrases '*Yahoo bought Overture*' and '*Yahoo acquired Overture*', within the sentences '*Yahoo bought Overture back in 2003*' and '*Yahoo acquired Overture in September 2003*', respectively, share the same set of anchors {*X*='*Yahoo*', *Y*='*Overture*'}, suggesting that the templates '*X* buy *Y*' and '*X* acquire *Y*' might paraphrase each other. Obviously, this approach cannot be applied to any arbitrary sentence pair. Indeed, in the example above, the two sentences,

from which the rule is extracted, convey similar meanings. In general, two steps are usually performed within the sentence-alignment approach. The first step finds sentences that share the same lexical contexts, as in the example above. This operation is termed here as (monolingual) sentence alignment. The motivation behind sentence alignment lies in the fact that two sentences that describe a similar event or fact are likely to paraphrase each other. Such sentences may differ in their structure but have similar meaning, so that the differences in structure may correspond to paraphrase rules (*i.e.* the obtained structures can be interchangeably used in other sentences). After alignment has been performed, the second step identifies the specific structures that paraphrase each other within the sentences and extracts them as a paraphrase rule. We next describe the manifestation of these steps in prior work.

2.1.1 Sentence-alignment step

In order to identify aligned sentence pairs easily and with high accuracy, some types of corpora, which contain highly redundant information, have been used: monolingual parallel corpora, such as multiple translations of the same text (Barzilay and McKeown 2001; Ibrahim *et al.* 2003), aligned bilingual corpora (Zhao *et al.* 2008), and comparable corpora, such as corresponding articles from multiple news sources of the same time period (Shinyama *et al.* 2002; Pang, Knight and Marcu2003; Barzilay and Lee 2003; Dolan, Quirk and Brockett 2004; Sekine 2005). The prominent method for identifying pairs of aligned sentences within corresponding texts in comparable/parallel corpora is to find sentences with a significant number of identical words (Barzilay and McKeown 2001; Shinyama *et al.* 2002; Barzilay and Lee 2003; Ibrahim, Katz and Lin 2003; Dolan *et al.* 2004).

While providing a highly accurate set of paraphrases, thanks to highly accurate sentence alignment, the main drawback of using parallel or comparable corpora lies in their limited availability. There are rather few parallel electronic translations into the same target language and, while there is an increasing number of news sources available on the Web, processing only topically matched articles significantly reduces the corpus size (the same largely holds for aligned corpora in different languages). Barzilay and Lee (2003) reported ∼6,500 paraphrase pairs extracted out of a 9-MB corpus of Middle-East news articles, and Shinyama *et al.* (2002) reported 136 paraphrase pairs extracted out of ∼300,000 newspaper articles.

Another inherent problem in using these types of corpora is that they usually include texts from one specific domain, such as the news domain, and thus can provide only paraphrases that are used in that domain. For example, the verb 'abduct' is used in the meaning of 'kidnapping' in news. However, 'abduct' is mainly used in the meaning of 'a type of muscle movement' in medical domains. We therefore conclude that due to their limited availability in size and domain coverage, comparable corpora cannot be the sole resource for paraphrase acquisition.

Avoiding the use of parallel or comparable corpora, Glickman and Dagan (2003) developed statistical methods that match verb paraphrases within a regular corpus, measuring the level of overlap between the anchors that instantiate each template. They obtained several hundred verb paraphrases from a 15 million word corpus, providing results of a limited scale that suggest that much larger corpora are required for large-scale

rule-sets. In a similar fashion, Sekine (2005) extracts rules between templates based on shared pairs of Named-entity anchors. To increase coverage, his algorithm first cluster templates with the same verb, where the anchor-set of the cluster is the disjunction of the anchors of its members. Hence, the algorithm identifies paraphrase relations between clusters of templates. The results are promising in terms of precision (around 80% on average), but were focused only on financial relations related to acquisitions, mainly for the verbs 'buy', 'acquire' and 'purchase', with no measure of coverage. To improve accuracy, Pekar (2006) learns rules only between templates related by local discourse. This algorithm relies on discourse information as a reliable link between related templates. It extract pairs of verb templates as candidates for an entailment rule if they share an argument instantiation in occurrences within the same paragraph. For example, from the two consecutive sentences '*Mary bought a house*' and '*The house belongs to Mary*', the candidate entailment relations {'*X* buy', 'belong to *X*'} and {'buy *X*', '*X* belong'} are extracted. However, information from different documents in the corpus is ignored, drastically limiting the number of candidate rules that may be extracted from the given corpus.

In search for a larger corpus in which many sentences can be aligned, the Web comes up as a natural candidate. It contains a great variety of expressions but also a lot of redundant information which is needed in order to find paraphrases. However, an exhaustive processing of the Web is not feasible and so sentence-alignment methods that were used in the above studies could not be utilized. Duclaye, Yvon and Collin (2002) and Ravichandran and Hovy (2002) attempted a 'selective' approach for learning paraphrases from the Web. Their methods start with a few seed examples of known anchor-sets that are manually provided for a target meaning. The choice of an anchor-set is such that all phrases within sentences containing a seed anchor-set are likely to express a single meaning. For example, the seed anchor-set {*X*='*Mozart*', *Y*='*1756*'} is given as input in order to find paraphrases for the semantic relation '*X* born in *Y*'. Web search engines are used to find occurrences of the input anchor-sets, resulting in sentences that are supposed to contain semantic relations equivalent to the target semantic relation (*e.g.* '*X* born in *Y*'). Templates that express the relations between *X* and *Y* in the retrieved sentences are then extracted and statistically ranked, *e.g.* 'born in *Y*, *X*' and '*X* born on *Y*' from within the sentences '*Born in 1756, Mozart began composing at the age of five*' and '*Mozart was born on 1756 in Salzburg, Austria*'. Both Duclaye *et al.* (2002) and Ravichandran and Hovy (2002) conducted qualitative tests over a dozen or so examples, aiming more for precision (finding correct paraphrase pairs) than for coverage. They did not provide quantitative results or analysis, but only output examples of a few learned paraphrases. Duclaye *et al.* report an average precision of 66.6% when only the 10% highest ranking templates are retained and an average precision of 42.9% when the 44% highest ranking templates are retained.

The main limitation of the above Web-based approaches is the requirement for at least one good input anchor-set per target meaning. Preparing a list of anchor-sets that provide a reasonable coverage of different paraphrases for all possible semantic relations in broad domains would be a huge task. In this work, we adopt the Web as our textual resource, due to its huge size, its heterogeneous content, and the amount of redundant information, which may help detect many entailment rules for different domains. However, our sentence-alignment method identifies anchor-sets automatically from the Web in an unsupervised manner, overcoming this limitation in prior work over the Web.

2.1.2 Rule extraction step

The output of the first step of the sentence-alignment approach consists of pairs of sentences which are assumed to contain phrases that express the same meaning. From these pairs, the structures of the paraphrase rules should be extracted. Some methods extract only pairs of constant lexical units (without variables) that paraphrase each other (Barzilay and McKeown 2001; Glickman and Dagan 2003). Yet, many paraphrases cannot be expressed by static lexical substitutions, for example '*X* prevent *Y*⇔*X* is for *Y* prevention'. In order to allow a more general and flexible paraphrase representation, different methods provide, as paraphrases, pairs of templates, consisting of constant terms and variables, which can be placed anywhere in the text fragment. The templates are considered substitutable under the same variable instantiations.

Previous work extracts two kinds of templates: linear templates, which are represented as sequences of words and variables, and lexical–syntactic templates, which are represented by parse sub-graphs where nodes can be constant words or variables. Several approaches were taken to identify linear templates in corpora. Sekine (2005) extracted the words between two named entities as candidate linear templates. Ravichandran and Hovy (2002) extract linear templates by finding repeated text fragments with variables using suffix trees, within variable-enhanced sentences. Zhao *et al.* (2008) extract linear templates in Chinese that correspond to English parse sub-trees in an aligned bilingual corpus. Two instantiated templates that correspond to the same instantiated English sub-tree are considered paraphrasing. More complex types of structures are learned in Barzilay and Lee (2003) and Pang *et al.* (2003). They learn linear paraphrases that are represented as finite state automata. The different paths in an automaton represent different possible paraphrases generated for the same event or fact. The structures represented by automata are usually sentence bound and describe a quite specific type of event (such as a terrorist attack), thus not capturing the more basic (or 'atomic') templates that can be combined to form larger, sentence-wide, transformations. On the other hand, shorter linear templates cannot easily represent long distance syntactic relations between words and variables in the template. For example, the lexical–syntactic template '$X \xleftarrow{subj}$ buy $\xrightarrow{prep}$ from $\xrightarrow{mod} Y$' hold a non-consecutive relation between 'buy' and *Y*, which is difficult to express concisely with linear templates. Thus, in this work, we follow the lexical–syntactic representation of templates described next.

Several studies extract lexical–syntactic templates, represented by parse sub-graphs. These templates are extracted based on the syntactic analysis of sentences, and are more general than linear templates since they can capture more distant relations in a sentence. Lin and Pantel (2001), Shinyama *et al.* (2002), Ibrahim, Katz and Lin (2003), Pekar (2006) and Shinyama and Sekine (2006) extract as templates syntactic paths from sentence parse graphs, where the nodes at the two ends of a path may be variables. For example, the path templates '$X \xleftarrow{subj}$ acquire $\xrightarrow{obj} Y$', '$X \xleftarrow{subj}$ acquire $\xrightarrow{prep}$ in $\xrightarrow{mod} Y$' and '$X \xleftarrow{obj}$ acquire $\xrightarrow{prep}$ in $\xrightarrow{mod} Y$' are extracted from '*Yahoo acquired Overture in September 2003*'. Following the sentence-alignment approach, paraphrase relations between different paths are identified when common anchors (context words) instantiate the template variables in different sentences (Shinyama *et al.* 2002; Ibrahim *et al.* 2003; Shinyama and Sekine 2006). While being more general than linear templates, limiting the template structure only

Table 2. *Example for feature vectors, one for each slot, constructed by the DIRT algorithm for two templates (scores are omitted for brevity). A significant amount of features are shared between the two templates (in bold), indicating their semantic similarity*

X find a solution to *Y*		*X* solve *Y*	
slot *X*	slot *Y*	slot *X*	slot *Y*
commission	strike	**committee**	**problem**
committee	**crisis**	clout	**crisis**
government	**problem**	**government**	mystery
legislator	budget deficit	petition	woe
sheriff	dispute	**sheriff**	murder

to paths between two nodes in a parse graph does not capture all possible paraphrases. For example, '*X* accuse *Y*' and '*X* call *Y* indictable' are paraphrases, but the second template is not a syntactic path between *X* and *Y*.

Sudo, Sekine and Grishman (2003) present a more general approach with respect to template structure, considering any sub-graph of a sentence dependency parse graph as a template candidate. This algorithm can learn any paraphrase that can be described by a connected parse sub-graph. However, its processing time increases exponentially with the size of the parse graphs it processes, because it extracts and evaluates every connected sub-graph in the sentence parse. In our work, we propose an algorithm for learning a general lexical–syntactic template structure, which is time and space efficient. Unlike Sudo et al, our algorithm does not test every sub-graph but rather identifies the most general repeated structure that is supported by the data.

2.2 The distributional similarity approach

Lin and Pantel (2001) present a notably different approach than sentence alignment, which relies on matching separately single anchors instead of matching sets of multiple common anchors in different sentences. Their method relies on the Distributional Hypothesis, which states that words that occur in similar contexts tend to have similar meanings. Instead of applying the Distributional Hypothesis to words, they apply it to paths in dependency parse graphs. Their algorithm, called *DIRT*, constructs a feature vector for each end point (variable slot) of each possible path in a parsed local corpus, where the features are the words that fill the slot in the different occurrences of the path in the corpus. Two path templates are marked as semantically related if they have similar vectors for both path ends. Table 2 presents an example for this process: two templates whose semantic similarity is indicated by the shared features within their corresponding feature vectors, which were constructed from a corpus. The example demonstrates the difference between DIRT and alignment-based algorithms, as DIRT decouples the instantiations of the *X* and *Y* slots, and does not consider the complete template instantiations.

Matching of single anchors, relying on the general distributional similarity principle, does not require repeated occurrences of sets of multiple anchors as do the other methods. Consequently, a much larger number of paraphrases can be found in a regular medium-size corpus. Lin and Pantel (2001) have constructed, from their corpus, a large knowledgebase

containing several million paraphrase rules. Indeed, DIRT is the most used algorithm for lexical–syntactic entailment rules in entailment engines (Giampiccolo *et al.* 2007). Yet, this method also suffers from certain limitations:

1. The structure of the templates that may participate in a rule is pre-defined, and was limited to paths by Lin and Pantel (2001).
2. Accuracy seems more limited due to the weaker notion of similarity, compared to sentence alignment based on the similarity of larger contexts.
3. As with previous methods that process a given local corpus, the coverage is limited to the scope and particular domains of the corpus.

To conclude, prior work for automatically learning entailment rules from corpora is limited in several respects. First, it appears that learning paraphrases from given local corpora as the only resource is not sufficient to create a broad coverage knowledgebase of paraphrases for various domains. On the other hand, the Web is a huge promising resource, but earlier Web-based methods suffer serious scalability constraints, as they require a supervised form of input (the anchor-sets). Furthermore, unlike algorithms that learn linear paraphrases, which use efficient algorithms for finding repeated generalized templates in the data, algorithms that learn lexical–syntactic paraphrases either use predefined structures for templates or exhaustively search through all possible structures, which is not scalable. In our work we attempt to address both the resource scalability issue, by providing an unsupervised algorithm that learns rules from the Web, and the template structure issue, by providing an algorithm that identifies the most general repeated lexical–syntactic structures, based on the given data.

3 The TEASE algorithm: unsupervised acquisition of entailment relations

In this section, we first discuss the goals we had in mind when designing the TEASE algorithm. We then present the outline of the algorithm and the key linguistic structures that are addressed by it: characteristic anchor-sets.

3.1 Goals and task

We address the task of learning entailment relations for a given lexical–syntactic input template I, *e.g.* the relations learned for 'X recover Y' in Table 1. Following our analysis of previous work, we aim at combining three complementing goals in order to scale up learning of entailment relations. First, we want to exploit the Web as our textual resource. Beyond being the largest existing corpus, it is highly heterogeneous, compared to local corpora, and it is also extremely redundant. These aspects provide a reliable starting point both for discovering entailment rules that might not appear in smaller corpora and for discovering rules for different meanings of the input template.

Our second goal is to overcome the scalability limitations of previous work when using the Web. These studies required an expensive manual input consisting of examples of valid contexts (anchor-sets) for the input template. We aim at identifying such anchor-sets in a completely unsupervised manner. To obtain wide coverage of entailment rules, we still need an initial list of input lexical–syntactic templates. However, constructing such a list is

easier than providing lists of anchor-sets for each input template. For example, an extensive list of verbs and noun phrases (both single and multiword phrases) can be automatically derived from a broad coverage lexicon or dictionary, such as WordNet (Miller 1995). The syntactic structure of these lexical phrases can be automatically constructed, for example by using sub-categorization frames described in WordNet or VerbNet (Kipper, Dang and Palmer 2000).

Our third goal aims at keeping template structures as general as possible. In particular, our algorithm efficiently learns template structures as syntactic trees without limitations on their configuration. This is necessary since in many cases templates are trees with more branches than just simple paths between the variables. For example, the template 'interaction between X and Y', which is not a simple tree path, is a valid paraphrase of the relation 'X interact with Y'. Another example is 'X take Y into custody' as a paraphrase of 'X arrest Y'.

Given the input template I, the output of the TEASE algorithm is a ranked list of templates $\{T_i\}$ where each template T_i is candidate to be in an entailment relation with I. The higher the rank of a template, the more confident the algorithm is that $\langle I, T_i \rangle$ is an entailment relation. An example of TEASE output for the input template 'X finish Y' is the list $\{T_1 =$ 'X end Y', $T_2 =$ 'X complete Y', $T_3 =$ 'X win Y'$\}$. The identification of the entailment direction between the input template and each of the output templates (*i.e.* understanding whether the direction is '$I \Leftrightarrow T_i$', '$I \Rightarrow T_i$' or '$T_i \Rightarrow I$') is beyond the scope of this article.

3.2 The TEASE algorithm outline

To present the TEASE algorithm outline, we first define a key linguistic structure: the anchor-set. A template T is *matched* in a sentence s if T is embedded in the parse tree of s. We then state that 'T is matched in s' or, equivalently, that 's is matched by T'. For example, the template 'X acquire Y' is matched in the sentence '*HSBC will acquire a stake in UTI Bank*'. An *anchor-set* is a set of terms that appear in the sentence together with the given template. In the above example, two possible anchor-sets are {X='*HSBC*', Y='*stake*'} and {X='*HSBC*', Y='*stake*', C_1='*UTI Bank*'}. We term as *slot anchors* the anchors that instantiate the template variables in the sentence, and they are labeled with their corresponding template variables. For example, 'HSBC' and 'stake' are slot anchors in the above example. The rest of the anchors, termed *context anchors*, appear in the sentence as context words, *e.g.* 'UTI Bank' in the above example. They are labeled with $C_i, 1 \leq i \leq N$, where N is the number of context anchors in the anchor-set. We say that an anchor-set is *minimal* if it contains only slot anchors. We also say that a template is instantiated by an anchor-set when the template's variables are replaced by the corresponding slot anchors of the anchor-set.

Usually, expressing a concrete event or fact requires a specific anchor-set AS together with a template T, which AS instantiates. For example {X='*BBC*', Y='*Tony Blair*'} describes different events when it instantiates 'X interview Y' and when it instantiates 'X criticize Y'. Yet, some anchor-sets hold the property of identifying the specific event of fact without the need to specify the template T. An example for such an anchor-set with respect to the template 'X compose Y' is {X='*William Shakespeare*', Y='*154*

For each input template I:

1. ASE: extract characteristic anchor-sets for the input template
   (a) Construct a sample corpus for the input template by retrieving sentences containing it from the Web.
   (b) Extract candidate anchor-sets from the sentences in the sample corpus.
   (c) Filter out candidate anchor-sets that fail certain criteria.
2. TE: extract templates
   (a) Construct a sample corpus by retrieving sentences containing the anchor-sets extracted in the ASE phase.
   (b) Extract repeated sub-structures in the sample corpus to be template candidates T_i.
3. Template Ranking: rank each extracted template T_i according to the confidence level in the validity of the entailment relation $\langle I, T_i \rangle$.

Fig. 1. The TEASE algorithm outline.

sonnets'}. Sentences containing this anchor-set, *e.g.* '*William Shakespeare penned 154 sonnets*', most likely refer to the lifetime achievement of Shakespeare in sonnet writing. Thus, templates instantiated by this anchor-set, *e.g.* '*X* pen *Y*' in the above example, would most likely participate in an entailment relation with '*X* compose *Y*'. We term such anchor-sets as *characteristic* of T. Following, in the TEASE algorithm, our goal is, given an input template I, to find characteristic anchor-sets for I, which will enable us to discover new templates that most likely participate in an entailment relation with I.

After having defined anchor-sets, we can outline the TEASE algorithm. For an input template I, the TEASE acquisition method consists of three phases, as outlined in Figure 1: the ASE phase (Section 4) automatically finds in the Web a substantial number of characteristic anchor-sets for I; the TE phase (Section 5) utilizes the acquired anchor-sets to find other templates, which are suggested as participating in an entailment relation with the input template; finally, TEASE ranks the output templates (Section 5.3).

We note that like other algorithms that iterate between discovering related templates and extracting their anchor-sets (Ravichandran and Hovy, 2002; Duclaye *et al.* 2002; Pantel and Pennacchiotti, 2006), our algorithm also lends itself to a bootstrapping scheme. In this scheme, more templates are found based on the current list of anchor-sets, and then more anchor-sets are retrieved based on the new list of templates, and so on. In the next two sections, we detail in depth the two phases of the TEASE algorithm. In our experiments, we then analyze in depth only the first iteration of our algorithm over these two phases (Section 7), which is the focus of this article; but we also demonstrate how the bootstrapping scheme can be applied to improve the performance of the TEASE algorithm (Section 7.2).

4 ASE: anchor-set extraction

Finding characteristic anchor-sets, based solely on the input template, is a difficult task. Only a very small portion of all anchor-sets that instantiate the input template occurrences in sentences are characteristic. Most anchor-sets are not characteristic, that is, they occur with many predicates that do not necessary entail each other. For example, {*X*='*Brown*', *Y*='*Blair*'} is not a characteristic anchor-set for the input template '*X* accuse *Y*', since it

Table 3. *Examples of correct characteristic anchor-sets learned for several input templates. X and Y are the slot anchors. C_1 is a context anchor*

Input template	Learned anchor-sets
X establish *Y*	{*X*='*epa*', *Y*='*national emission standard*', C_1='*asbestos*'} {*X*='*canada agricultural products act*', *Y*='*review tribunal*'} {*X*='*school district*', *Y*='*breakfast program*'} {*X*='*federal government*', *Y*='*conservation corps*'} {*X*='*erisa*', *Y*='*minimum standards*'} {*X*='*constantine*', *Y*='*new rome*'}
X write *Y*	{*X*='*laurie*', *Y*='*numerous songs*'} {*X*='*lewis carrol*', *Y*='*alice's adventures*'} {*X*='*plato*', *Y*='*detailed account*', C_1='*atlantis*'} {*X*='*mendelssohn*', *Y*='*incidental music*'} {*X*='*shakespeare*', *Y*='*great tragedies*'} {*X*='*thomas malthus*', *Y*='*essay*'}
X calculate *Y*	{*X*='*katz equation*', *Y*='*membrane potential*'} {*X*='*eratosthenes*', *Y*='*circumference*'} {*X*='*nernst equation*', *Y*='*equilibrium potential*'} {*X*='*language model*', *Y*='*probabilities*'} {*X*='*following table*', *Y*='*annual cost*'} {*X*='*acos*', *Y*='*arc cosine*'}

could also be found in sentences such as '*Gordon Brown met Tony Blair to discuss the university tuition fees*' and '*Brown praised Tony Blair as a stalwart leader in the fight against terrorism*', for which the instantiated templates, '*X* meet *Y*' and '*X* praise *Y*' , do not participate in an entailment relation with '*X* accuse *Y*'.

Previous methods that identified characteristic anchor-sets in a regular corpus cannot be utilized by Web-based methods because they require processing the full corpus (Lin and Pantel 2001; Sekine 2005; Pekar 2006). This is why prior Web-based methods relied on manually given characteristic anchor-sets, as described in Section 2.1.1. For automatic unsupervised acquisition we need refined criteria that identify the relatively few characteristic anchor-sets within a sample of sentences containing the input template.

Given an input template I, the ASE algorithm outputs a list of anchor-sets, denoted $AS(I)$, that are assumed to be characteristic for the input template. Table 3 displays examples of output anchor-sets for several input templates. Figure 2 presents the main steps of the ASE algorithm, which include the construction of a corpus from the Web, consisting of sentences containing the template I, and identification and extraction of characteristic anchor-sets from the retrieved corpus. We next describe in detail each of the three steps presented in Figure 2.

4.1 Step 1: constructing a sample corpus

This step builds a sample corpus S, where each sentence $s \in S$ is matched by I. It consists of two sub-steps. The first sub-step utilizes a Web search engine to retrieve distinct

For an input template I:

1. Construct a sample corpus that consists of sentences containing I:
   (a) Retrieve sentences from the Web using a query containing the template's words.
   (b) Retrieve more sentences from the Web using refined queries, based on the sentences retrieved at step (1.a).
2. Extract characteristic anchor-set candidates from the constructed corpus:
   (a) Extract one minimal anchor-set, containing only the slot anchors, from each sentence in the sample corpus.
   (b) Extract one more anchor-set from each sentence, containing one context anchor in addition to the slot anchors, if possible.
3. Filter out candidates that fail certain criteria:
   (a) Applying statistical thresholds over individual anchor-sets.
   (b) Filtering anchor-sets that are redundant or inconsistent relative to other anchor-sets.

Fig. 2. The ASE algorithm outline.

sentences containing all the words in I. The collected sentences are then parsed using Minipar. Finally, a parsed sentence s is added to S only if I is (syntactically) matched in s. Instead of downloading the full documents and searching for sentences containing I, we analyze only the document snippets returned by the search engine, in order to increase the processing speed. We utilize the Yahoo search engine API[4], retrieving up to the top 1,000 snippets (the API limit) for the query containing the words in I. Each snippet is then split into sentences using MXTERMINATOR (Reynar and Ratnaparkhi 1997) prior to parsing it.

The second sub-step attempts to retrieve additional sentences that are matched by I, adding them to S. We aim to increase the size of S since the proportion of characteristic anchor-sets in a given corpus is small. This sub-step first identifies terms and phrases that are statistically associated with I in S, and will be used as query expansion terms in additional queries. For this purpose, all noun phrases np in S that are not on a stop list are extracted and sorted by:

$$tf \cdot idf(np) = sentf_S(np) \ \log\left(\frac{N_{web}}{docf_W(np)}\right)$$

where $sentf_S(np)$ is the number of sentences in S containing the noun phrase np, $docf_W(np)$ is the estimated number of documents containing np as returned by the search engine, and N_{web} (the total amount of Web documents) is estimated by 10^{10} (Baeza-Yates and Raghavan 2010). Finally, the top 20 noun phrases are chosen for the expansion process. The chosen noun phrases have a relatively strong association with the input template, due to the fairly high $tf \cdot idf$ score, and are likely to appear in sentences that demonstrate typical usages of the input template. For example, for the input template '*X* prevent *Y*', the noun phrase '*American Dental Association*', which indicates sentences on health topics, was chosen. For each chosen noun phrase, the algorithm re-queries the Web for distinct sentences that contain the chosen phrase in addition to all the words in the input template.

[4] http://developer.yahoo.com/search/

It then parses the sentences of the retrieved snippets and adds to S only sentences matched by the input template, as in the first sub-step.

4.2 Step 2: extracting anchor-set candidates

This step extracts anchor-set candidates for the input template from the collected sample corpus S, constructing the list of anchor-set candidates $ASC(S)$. Consisting of two sub-steps, the first sub-step extracts for every sentence $s \in S$ one minimal anchor-set candidate as_s. The algorithm extracts the noun phrases (the head and its modifiers) of the variable instantiations of I in s to be the slot anchors of as_s. For example, the minimal anchor-set $\{X=$'*vitamin d*', $Y=$'*fractures*'$\}$ was extracted for the input template 'X prevent Y' from the sentence '*vitamin d may prevent some fractures in elderly people*'. An anchor-set is not extracted if one of its anchors is on a stop list. For example, the anchor-set $\{X=$'*he*', $Y=$'*Lincoln*'$\}$ is not extracted from '*He killed Lincoln*' for the template 'X kill Y' because 'he' is on our stop list. The algorithm associates each slot anchor with the corresponding variable in I, to be used later in the TE phase (Section 5) when aligning sentences containing different anchor-sets.

The minimal anchor-set may be not sufficiently characteristic. To improve its specificity, the second sub-step attempts to extract one more anchor-set candidate from each sentence s by expanding the minimal anchor-set candidate, with one additional context anchor from s. The context anchor chosen is the highest $tf \cdot idf$ scoring noun phrase in the sentence,[5] if exists, where noun phrases that have already been chosen as slot anchors are ignored. For example, the context anchor 'elderly people' was chosen from the previous example to expand the minimal anchor-set into $\{X=$'*vitamin d*', $Y=$'*fractures*', $C_1=$'*elderly people*'$\}$. We do not attempt to add more than one context anchor because our experiments showed that most anchor-sets that contain two or more context anchors are too specific and typically would not occur with other templates.

4.3 Step 3: filtering anchor-set candidates

This step filters out anchor-set candidates that are unlikely to be characteristic, based on certain statistical criteria. The indicative feature of a characteristic anchor-set is that it should be specific enough. We have investigated different statistical filters and found that relatively simple criteria produced the best results for discriminating between characteristic and non-characteristic anchor-sets. For each anchor-set candidate $AS_i \in ASC(S)$, the following two filtering criteria are applied.

Absolute frequency threshold (filter 1). AS_i is maintained only if $docf_W(ASi)$ is lower than a threshold $MaxAbsF$ (set to 15,000 in our experiments). If the anchor-set appears in too many documents it usually means that the anchor-set is too generic and not sufficiently characteristic.

[5] The $tf \cdot idf$ score used here is the same as in Step 1.

Conditional probability threshold (filter 2). AS_i is maintained only if it holds that $Pr(I(AS_i)|AS_i)$ is higher than a threshold $MinRelPr$ (set to 0.666 in our experiments), where $I(AS_i)$ is the input template instantiated with AS_i. $Pr(I(AS_i)|AS_i)$ is estimated as:

$$Pr(I(AS_i)|ASi) \approx \frac{docf_W(I(AS_i))}{docf_W(AS_i)} \quad (1)$$

where $docf_W(I(AS_i))$ is estimated by $docf_W(I \wedge AS_i)$, the number of documents on the Web containing the terms of both the input template and all the anchors in AS_i. This approximation is done in order to make processing time feasible, and it requires just one query to the Web without additional post-processing.

The conditional probability expresses a qualitative value of the relation between a characteristic anchor-set and the input template I. Since the Web is highly redundant, we expect that if I captures well the semantic relation between the anchors of AS_i then it will be used in several occasions to form sentences with AS_i. Too low conditional probability indicates that it is rather coincidental to find I in a sentence containing AS_i. Hence, the input template does not capture prominently enough the relation between the slot anchors of AS_i.

It should be noted that we have chosen to use conditional probability as a filtering criterion over the common point-wise mutual information $pmi(I, AS_i) = \frac{Pr(I(AS_i))}{Pr(AS_i) \cdot Pr(I)}$. This is because our measurement is not symmetric with respect to the anchor-set and the input template. It is valid for the input template to be joined with many different anchor-sets, either under the same sense or different senses of the template. On the other hand, to be characteristic, an anchor-set should capture only one core meaning that correlates well with one of the input template's senses.

In addition to filtering each anchor-set based on its own qualities, we found that several relations between different anchor-set candidates may result in either learning erroneous templates in the subsequent TE phase or retrieving redundant information from the Web. We handle these cases using the following three filters.

Choosing between minimal and expanded anchor-sets (filter 3). If, for a given sentence, both the minimal anchor-set and the expanded anchor-set are retained, the minimal anchor-set is discarded. The anchor-set candidate with more anchors is kept since it is more characteristic for the described event.

Expanded anchor-set filtering (filter 4). Some anchor-sets contain anchors that are an expanded version of noun-phrase anchors in other anchor-sets. For example, in {X='*Los Angeles Lakers*', Y='*Boston Celtics*'} 'Los Angeles Lakers' is an expanded version of 'Lakers' in {X='*Lakers*', Y='*Boston Celtics*'}. Anchor-sets that are an expanded version of other anchors-sets in $ASC(S)$ are filtered out, since they would only retrieve sentences that are already retrieved by the shorter anchor-set. The shorter anchor-set candidate is preferred because otherwise there could be several expanded versions of the same anchor-set, which return identical or nearly identical sentences. This in turn contributes to noise for the subsequent TE phase. A special case is duplicate anchor-sets, where only one of them is retained.

Permutation filtering (filter 5). Some anchor-sets are permutations over the variables of other anchor-sets. For example, for the input template '$X \xleftarrow{subj}$ beat $\xrightarrow{obj} Y$', the sentences:

- '*the Los Angeles Lakers beat the Boston Celtics*'
- '*the Boston Celtics beat the Los Angeles Lakers*'

produce the anchor-sets:

- $AS_1 \equiv \{X=$'*Los Angeles Lakers*', $Y=$'*Boston Celtics*'$\}$
- $AS_2 \equiv \{X=$'*Boston Celtics*', $Y=$'*Los Angeles Lakers*'$\}$

which are permutations of each other.

Unless the predicate described in the input template is a symmetric relation between the variables, as in 'X meet with Y', permutation anchor-sets indicate that there is no consistent semantic relation between the slot anchors of the anchor-set. Therefore, templates that are instantiated by these anchor-sets have no consistent semantic interpretation. In our example, from the sentence '*the Los Angeles Lakers lost to the Boston Celtics*' both the template '$X \xleftarrow{subj}$ lose to $\xrightarrow{obj} Y$', instantiated by AS_1, and the template '$Y \xleftarrow{subj}$ lose to $\xrightarrow{obj} X$', instantiated by AS_2, can be extracted. Thus, the template in the sentence is interpreted both as a synonym and as an antonym for the input template 'X beat Y'. For most predicates, which express an asymmetric relation between their arguments, this is an undesired behavior that should be avoided. Hence, all anchor-sets that are permutations of each other are discarded.

Finally, the list $ASC(S)$, containing the remaining anchor-set candidates, is provided as $AS(I)$, the output of the ASE phase. Table 3 provides examples of anchor-sets extracted by the ASE phase for several input templates. All anchor words are lower cased as part of the extraction. Most anchor-sets are minimal and do not contain an additional context anchor. There are many good characteristic anchor-sets extracted by the ASE phase, as can be seen in Table 3. However, some extracted anchor-sets are too general, for example due to badly estimated conditional probability $Pr(I(AS_i)|ASi)$, which falsely indicated the input template as strongly related to an anchor-set. This happens due to the inaccurate estimations for search-result counts within search engines. In addition, other anchor-sets are unrelated to the template, and were extracted due to parse errors. Table 4 provides some examples for such invalid anchor-sets.

5 TE: template extraction

Once anchor-sets are extracted for the input template I in the ASE phase, the TEASE algorithm next tries to extract templates that are in an entailment relation with I. The Template Extraction (TE) phase receives as input $AS(I)$, the list of anchor-sets that are assumed to be characteristic for the input template. The output of this phase is a list of syntactic templates, denoted $TE(I)$, which are candidates for holding an entailment relation with the input template.

The TE phase is performed by first constructing a sample corpus S of sentences from the Web containing the input anchor-sets. Then, a set of lexical–syntactic templates, which are repeated in S, are extracted from S via a novel algorithm, which we call *General Structure*

Table 4. *Examples of incorrect characteristic anchor-sets extracted by the ASE phase*

Input template	Incorrect anchor-sets
X establish *Y*	{*X*='*definition screen*', *Y*='*template*'} {*X*='*compiler*', *Y*='*exception handler*'} {*X*='*muslims*', *Y*='*khaleefah*'}
X write *Y*	{*X*='*reporters*', *Y*='*15 times*'} {*X*='*moses*', *Y*='*ten commandments*'} {*X*='*screenplay*', *Y*='*short novel*'}
X calculate *Y*	{*X*='*following table*', *Y*='*annual cost*'} {*X*='*spreadsheet*', *Y*='*following items*'} {*X*='*estimator*', *Y*='*sample variance*'}

Learning (GSL). The GSL algorithm builds a *Compact Graph Representation* (CGR) of S and then extracts *Least General Maximal Generalization (LGMG)* templates from this representation. Finally, the templates extracted from the GSL algorithm are ranked. We next describe each of these steps in detail.

5.1 Constructing a sample corpus for the input anchor-sets

The first TE step constructs a sample corpus S of parsed sentences[6] containing anchor-sets from $AS(I)$. For each anchor-set $AS_i \in AS(I)$ a Web search engine is utilized to retrieve a set of distinct sentences, where each sentence contains all the anchors in AS_i. The sentences are taken from the snippets retrieved for a query containing the quoted anchor-set terms. Each retrieved sentence s is parsed using Minipar, whose output represents the syntactic dependency structure of the sentence. Then, every slot anchor in AS_i is identified in the parsed sentence and replaced with its corresponding variable node label of the input template. The replacement by variable names enables the identification of repeated parse sub-graphs in sentences that were retrieved for different anchor-sets. Finally, the parsed sentence with replacements, which we denote by $P_{dep}(s)$, is associated with AS_i and added to S.

For example, the sentences '*Aspirin stops first heart attack*' and '*vitamin C stopped hip dysplasia in dogs*' were retrieved, respectively, for {*X*='*aspirin*', *Y*='*heart attack*'} and {*X*='*vitamin C*', *Y*='*hip dysplasia*'}, two anchor-sets in $AS(X \xleftarrow{subj} \text{prevent} \xrightarrow{obj} Y)$. Replacing the slot anchors with the corresponding variable names obtained the parsed sentences '$X \xleftarrow{subj} \text{stop} \xrightarrow{obj} Y \xrightarrow{post} \text{first}$' and '$X \xleftarrow{subj} \text{stop} \xrightarrow{obj} Y \xrightarrow{mod} \text{in} \xrightarrow{pcomp-n} \text{dogs}$', enabling the identification of the repeated sub-graph '$X \xleftarrow{subj} \text{stop} \xrightarrow{obj} Y$'.

In our experiments, we found that retrieving sentences for anchor-sets was the most time-consuming step of the TE phase. In order to decrease the processing time, not all anchor-sets in $AS(I)$ were processed. Instead, $AS(I)$ was randomly sampled, without repetition,

[6] We use S here as well for notation simplification but emphasize that in this section S refers to the corpus generated at the *TE* phase and not to a single sentence, as used in the *ASE* phase.

for anchor-sets to process, until the number of sampled anchor-sets reached some limit (set to 600 in our experiments).

5.2 The general-structure learning algorithm

The *GSL* algorithm is the core of the TE phase, which efficiently learns templates from the corpus S. We first introduce two structures that are in the heart of the algorithm: *LGMG templates* and *CGR*. We then describe the GSL algorithm and discuss its efficient implementation.

5.2.1 Least general maximal generalization templates

Most prior work focused on extracting templates whose structure is a path between two variables (see Section 2.1.2). However, a path is only one of possible valid structures for a template, and limiting template structures only to paths may result in incorrect template learning. For example, in the upper point of Figure 4, extracting only path templates would yield 'X take Y' instead of 'X take Y into custody', whose structure is not a path but does specify the appropriate template given the data. We aim at the right generalization level in the template learning process: we want to learn templates that are general enough to cover as much empirical data as possible, but, on the other hand, we do not want to perform invalid generalizations. We note that in this article, our experiments and examples focus on templates with two variables, yet the GSL algorithm described in this chapter can also be applied to templates with more than two variables.

This problem has been widely studied in the machine learning literature. In particular, learning the appropriate level of generalization of structured objects was investigated in Inductive Logic Programming (ILP) (Bergadano and Gunetti 1995). A classical key concept within this paradigm is the least general generalization (LGG), which is defined as the least general structure that subsumes a given empirical set of structures (*e.g.* logical formulas) (Reynolds 1970; Kietz and Lubbe 1994; Markov and Pelov 1998). We propose here a similar rationale, captured by a notion of an *LGMG template*, as described below.

Definition

A *spanning template* is a connected parse sub-graph that contains all the slot variables from the input template.

The GSL algorithm learns spanning templates from the parse graphs in S. In this section, we consider the notion *template* as referring to spanning templates.

Definition

For each template t, we define a *support set*, denoted by $\sigma(t)$, which consists of all the occurrences of t in S. Each occurrence of t is a sub-graph, isomorphic to t, and belongs to some $s \in S$. We denote by $as(t)$ the different anchor-sets that instantiate t in S.

Considering the example in Figure 3, the template '$X \xleftarrow{subj}$ stop $\xrightarrow{obj} Y$' has a support set that includes two occurrences (in the parse graphs G_1 and G_2).

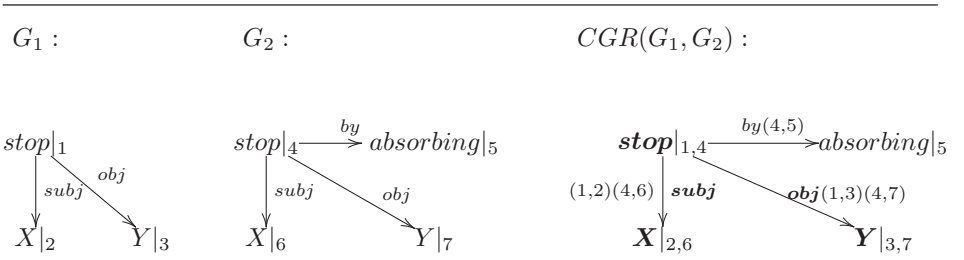

Fig. 3. Two parse trees and their CGR. The indexes on the nodes, *e.g.* $|_6$, denote vertex labels or sets of labels. The index pairs on the edges, *e.g.* $(1,2)(4,6)$, denote edge labels or sets of labels. Nodes and edges in bold indicate the LGMG template within the CGR graph.

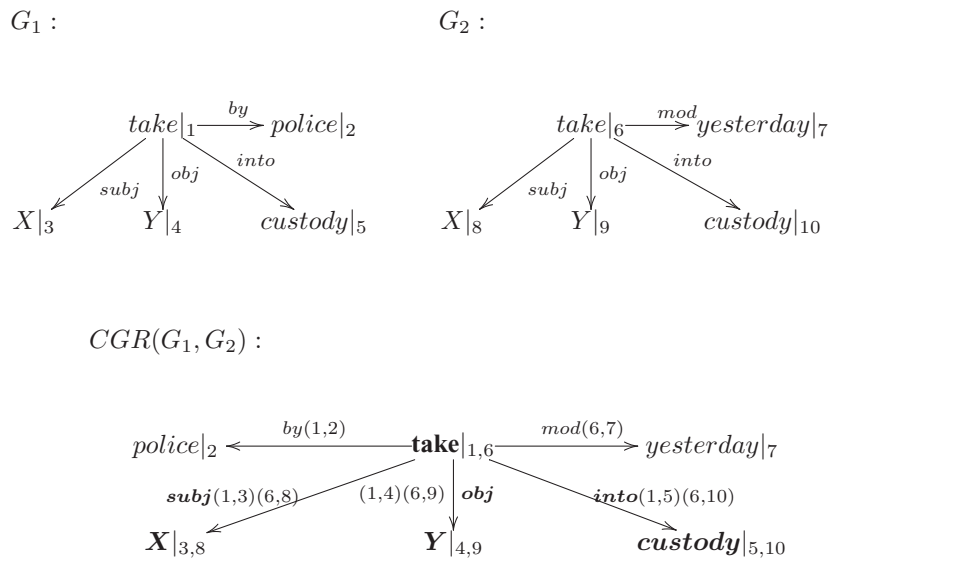

Fig. 4. Two parse trees and their CGR, where the LGMG template (bolded nodes and edges) in the *CGR* graph is not a path between the two variables.

Definition

A template t_1 *subsumes* another template t_2 if t_1 is more general than t_2, that is t_1 is strictly a sub-graph of t_2. This relation is denoted by $t_1 \prec t_2$. If $t_1 \prec t_2$, then $|\sigma(t_1)| \geq |\sigma(t_2)|$, since every occurrence of t_2 includes as a sub-graph an occurrence of t_1.

For example, in Figure 3 the template 'X stop Y' subsumes 'X stop Y by absorbing'.

We denote with $T(S)$ the set of all templates that have at least *minFreq* occurrences in S, *i.e.* $t \in T(S)$ iff $|\sigma(t)| \geq minFreq$. (In our experiments, we set $minFreq = 2$.) The subsumption relation defines a partial order over $T(S)$. Using this partial order, the template set $T(S)$ can be mapped onto a subsumption lattice similar to the lattices used in ILP (Markov and Pelov 1998). On the top of this lattice reside the *most general templates*, which are not subsumed (and therefore not preceded in the partial order) by any other

template in $T(S)$. Due to the requirement for spanning over all the input variables, the most general templates are the (locally) minimal spanning trees that connect the anchor variables (*i.e.* templates that cannot be further reduced without omitting a variable, *e.g.* the template '*X* stop *Y*' in Figure 3).

For each most general template t in $T(S)$, we consider all the templates it subsumes that have the same support set as t. We define the least general of these templates – the one that does not subsume another template with the same support set – as an LGMG template.[7] (t itself can be an LGMG template if it does not subsume any template with the same support set.) We note that for each most general template t in $T(S)$ there exists one and only one LGMG template. Otherwise, if there are two different LGMG templates t_1 and t_2 (each contains elements not appear in the other), their combination is yet another template t_3. t_3 is different than t_1 and t_2 (it contains elements from both templates), it is subsumed by both, and also share their support set, since if both templates appear in each sentence in the support set, their combination appears there too. Following, t_3 is the LGMG template and not t_1 and t_2.

For example, in Figure 3 the template '*X* stop *Y*' is an LGMG template, since it cannot be further expanded without decreasing the size of the support set. As another example, in Figure 4 the most general template is the path '*X* take *Y*'. However, the LGMG template would be '*X* take *Y* into custody', since the support set of this template is the same as the support set of the most general template by which it is subsumed.

The following formally defines LGMG templates:

Definition
A template $t \in T(S)$ is an *LGMG template*, if the following conditions hold:

Condition A – Maximal Generalization: $\forall t' \in T(S)\ s.t.\ t' \prec t,\ |\sigma(t)| = |\sigma(t')|$.
Condition B – Least General Generalization: $\forall t' \in T(S)\ s.t.\ t \prec t',\ |\sigma(t)| > |\sigma(t')|$.

Condition A – *Maximal Generalization* ensures that further generalizations of an LGMG template do not augment the support set and therefore we have a (locally) maximal generalization with respect to the empirical data (the support set). Notice that the maximal generalization introduced by condition A is not global and many templates in $T(S)$ may have this property.

Condition B – *Least General Generalization* ensures that all the templates that an LGMG template subsumes have smaller support sets, which means that LGMG templates cannot be specified further without decreasing their support set. Therefore, an LGMG template is the least general in each group of maximal generalization templates that are related through the subsumption relation.

5.2.2 Compact graph representation

The CGR data structure was introduced in Szpektor *et al*. (2004), where it was used to learn the structure of entailment relations. It was also used in Tanev and Magnini (2006),

[7] In this term, 'least general' refers to the template structure while 'maximal generalization' refers to the maximal size of the support set.

where it was used to extract syntactic features for ontology population. In Tanev (2007) an efficient CGR-based algorithm for syntactic pattern matching was described and exploited for relation extraction (RE).

The purpose of CGR is to represent a set of labeled graphs by one aggregate structure. In particular, in the context of Template Extraction, CGR represents the set of parsed sentences S. In this case, CGR is denoted as $CGR(S)$. In $CGR(S)$, vertices labeled with equal labels from all the graphs in S are merged into one generalizing vertex. Therefore, each vertex label in S appears just once in the CGR. Consequently, there is only one vertex for each slot variable. Moreover, all (derived) arcs with an identical label that connect equally labeled vertices in the same arc direction are merged in one aggregate arc in $CGR(S)$.

Figure 3 exemplifies two parse trees G_1, G_2 (obtained for the input template 'X prevent Y'), as well as their CGR. The slot variable nodes in all the graphs are named X or Y according to the variable in the input template that they are mapped to. The two vertices labeled with 'stop' in G_1 and G_2 are merged into one vertex in $CGR(S)$ with the same label; in the same way, the two arcs 'stop $\overset{subj}{\rightarrow}$ X' are merged into one arc in the CGR.

The goal of $CGR(S)$ is to facilitate detection of repeated connected components in S. To achieve that, each vertex in the graphs from S should be uniquely identifiable in $CGR(S)$. To this end, we assign to each vertex in the graphs from S a unique integer identifier (*id*) in S. For example, in Figure 3 the id of the vertex 'stop' in G_2 is 4. Using these ids each arc in S may also be represented uniquely through the ids of its initial and terminal vertices. For example, the arc ('stop', Y) in G_2 may be represented with the id pair (4,7).

To trace the original vertices and edges of the graphs from S within $CGR(S)$, we further annotate its vertices and edges with *index sets*. Each vertex in $CGR(S)$ has a corresponding index set that contains the ids of the vertices from S that are represented by this CGR vertex (have the same label). In Figure 3, the vertex 'stop' in the CGR has a corresponding index set $\{1,4\}$, since the vertices with ids 1 and 4 are represented by this CGR vertex. In the same way, each arc in $CGR(S)$ has a corresponding index set that contains the id pairs of the edges represented by this aggregate edge. In Figure 3, the arc 'stop $\overset{subj}{\rightarrow}$ X' in the CGR has a corresponding set of two id pairs $\{(1,2), (4,6)\}$, which indicates that arcs (1,2) and (4,6) are represented by this aggregate arc.

Based on the index sets, $CGR(S)$ contains the information needed to trace all shared connected components in S. Since isomorphic structures in S coincide in $CGR(S)$, we can efficiently find the occurrences of each repeated substructure in S. In the context of Template Extraction, we use this last property to efficiently trace repeated patterns that span over the anchor variables and to collect statistics about their occurrences.

5.2.3 GSL – an efficient algorithm for learning LGMG templates

Following the definition of LGMG templates and CGR, the GSL algorithm can be summarized as an algorithm that extracts LGMG templates from the CGR of the corpus S. Following, we outline its algorithmic steps, which are fully detailed in Section 5.2.4 and in the Appendix.

To first construct $CGR(S)$, GSL initializes it to be an empty graph. Then, for each graph $s \in S$, its vertices are first considered and next its edges. If a vertex v in s has the same

label as an already existing vertex v_{cgr} in $CGR(S)$, then the id of v is added to the index set of v_{cgr}. If this label is missing from $CGR(S)$, a new vertex that is labeled accordingly is added to $CGR(S)$ and the id of v is inserted into the index set of the new vertex. Arcs are considered in a similar way.

After $CGR(S)$ is constructed from the sample corpus S, GSL extracts all connected templates that span over all slot variable nodes such that there is no other spanning template that subsume them. Only the templates that occur at least *minFreq* times in S are considered. Such templates are termed here *most general spanning templates* (MGST). By definition an MGST template is a maximal generalization template. Graphically, an MGST template is a (locally) minimal spanning tree for the input variable nodes. Once extracted, every MGST template t' is further expanded to the maximal sub-graph t containing t' that has the same support set as t'.

As a result of this algorithm, each extracted template satisfies the following: (i) It cannot be expanded further while keeping the same number of occurrences. Therefore, condition B – *Least General Generalization* from the LGMG definition – holds for t. (ii) The template t has the same number of occurrences as the MGST template s that subsumes it. Since s subsumes also all other spanning templates that subsume t, also condition A – *Maximal Generalization* from the LGMG definition – holds. Therefore, t is an LGMG template. Since each LGMG template contains an MGST template as a sub-graph, and all MGST templates are iterated, the GSL algorithm extracts all LGMG templates in $CGR(S)$.

For example, in Figure 3 the minimal spanning tree 'X stop Y' is an MGST template. In Figure 4 'X take Y' is an MGST template. The template 'X stop Y' (Figure 3) cannot be further expanded and it is also an LGMG template. However, 'X take Y' (Figure 4) is expanded by GSL into 'X take Y into custody', which is the LGMG template in the graph.

5.2.4 *Empirical constraints and efficient implementation*

In our experiments, we extract templates which have at least two occurrences in S (setting $minFreq = 2$). This is because, following our algorithm, templates which appear only once would be expanded to the whole sentence in which they appear. In addition, empirical observations lead us to consider only templates in $T(S)$ with the following linguistic properties: (a) the template is a directed acyclic graph (DAG) with a single root; (b) the template's root coincides with the root of a (locally) minimal spanning tree between the variables; (c) the root is a lexical node (not a Minipar clause node). From our observations, we noticed that templates that do not have these properties are typically learned from incorrectly parsed sentences.

Given an input template with N variables, in order to find minimal spanning trees, the algorithm builds paths that end at each variable vertex in the CGR. Then, it constructs trees from any N paths ending in different variables, which have a common ancestor, generating spanning trees over the variable vertices. Next, each spanning tree is expanded into the largest DAG which appears as many times as the spanning tree. In all the algorithm stages, we use the index sets on the CGR vertices and edges to efficiently calculate the occurrences of the paths, trees and DAGs. Under certain assumptions, the number of operations necessary for the algorithm is estimated to be $O(|w| \cdot log|w|)$, where $|w|$ is the

Table 5. *Examples of correct templates learned for several input templates*

Input template	Learned templates	
X establish *Y*	*X* set *Y*	*X* promulgate *Y*
	X develop *Y*	*X* issue *Y*
	X create *Y*	*X* implement *Y*
	X found *Y*	*X* provide *Y*
	X enforce *Y*	*X* make *Y*
	X form *Y*	*X* launch *Y*
	X offer *Y*	*X* institute *Y*
	X release *Y*	*X* for the establishment of *Y*
X write *Y*	*X* who write *Y*	*X* produce *Y*
	X publish *Y*	*X* pen *Y*
	X compose *Y*	*X* create *Y*
	read *Y* by *X*	*X* 's *Y*
	Y attributed to *X*	*X* complete *Y*
	perform *Y* by *X*	*X* book of *Y*
	X writer of *Y*	*X* say in *Y*
	selected *Y* of *X*	*X* work include *Y*
X calculate *Y*	*X* determine *Y*	*X* measure *Y*
	X compute *Y*	*X* calculation of *Y*
	X give estimate of *Y*	*X* yield *Y*
	X return *Y*	*X* get *Y*
	X assess *Y*	*X* produce *Y*
	X generate *Y*	*X* according to *Y*
	X recalculate *Y*	*X* obtained from *Y*
	X work out *Y*	*X* evaluate *Y*

number of the words in the corpus. See the Appendix for detailed algorithm description and efficiency analysis.

The learned templates are provided as the output list $TE(I)$. Table 5 provides examples of templates extracted by the TE phase for several input templates.

5.3 Template ranking

The final step of the TE phase ranks the template candidates in $TE(I)$, extracted by the GSL algorithm. The target of the ranking is to indicate which of the candidates is more plausible to be correct: the higher the rank of a template, the more confident the algorithm is that the template participates in an entailment relation with the input template.

Our ranking method is based on the number of different anchor-sets and sentences supporting a template. The input is the list $TE(I)$ and the output is an ordered list $TEASE(I)$ of the templates $t_i \in TE(I)$. We sort the different templates first by the number of different anchor-sets that support each template, $|as(t_i)|$ and then by the number of the supporting sentences, $|\sigma(t_i)|$. This ranking method follows the motivation that since the anchor-sets extracted are intended to be characteristics, the more anchor-sets supporting a template t_i

the more it is likely that t_i is supported by a sufficient number of characteristic anchor-sets, which indicate thatt_i participates an entailment relation with I.

6 Evaluation methodology and setting

We next proceed to a comprehensive evaluation of the TEASE algorithm. Despite the many methods for automatic rule acquisition introduced in recent years, as described in Section 2, there has not been a common accepted framework for their evaluation. A task-based approach for evaluating automatically acquired rules is to measure their contribution to the performance of specific NLP systems, such as QA (Ravichandran and Hovy 2002) or IE (Sudo, Sekine and Grishman 2003; Romano *et al.* 2006). While measuring the impact of learned rules on applications is highly important, it cannot serve as the primary approach for evaluating acquisition algorithms, since NLP systems have many components that address multiple phenomena, and thus it is hard to assess the effect of a single resource in isolation.

Therefore, the predominant approach for evaluating the quality of rule acquisition algorithms has been by human judgment of the learned rules (Lin and Pantel, 2001; Shinyama *et al.* 2002; Barzilay and Lee 2003; Pang *et al.* 2003; Szpektor *et al.* 2004; Sekine 2005). In this evaluation scheme, termed here the *rule-based approach*, a sample rule is correct or not, *i.e.* whether they can think of reasonable contexts under which the rule holds. However, coming up with appropriate contexts is not an easy task – a judge may legitimately fail in thinking of a valid context that was caught by another judge. Indeed, only few earlier studies reported inter-judge agreement level for this kind of evaluation, and those that did reported rather low κ values (Carletta 1996), such as 0.54 (Barzilay and Lee 2003) and 0.55–0.63 (Szpektor *et al.* 2004). According to our experience, this approach turns out to be problematic because the rule correctness criterion is not sufficiently well defined and is hard to apply consistently.

These observations motivate a different approach, which is based on the evaluation of *instantiated* rules in context. Such *instance-based* methodology has been formalized and assessed in Szpektor *et al.* (2007). In another work, Pantel *et al.* (2007) also show improved agreement (0.72 κ) for manually judging the correctness of a rule by looking at examples for the rule's application. In our work, we follow the instance-based evaluation scheme and summarize it in Section 6.1 (the complete details are found in Szpektor *et al.* 2007). We then proceed to describing the corresponding experimental setting for the TEASE algorithm evaluation (Section 6.2). The evaluation results are presented and discussed in Section 7.

We note that while our main experimental setup is based on manual evaluation, we do think that a task-based evaluation can provide additional insights into the performance of our algorithm. We thus also tested the performance of TEASE on an RE evaluation setup, whose settings and results are presented in Section 7.2.

6.1 Instance-based evaluation methodology

In order to assess if a rule '$L \rightarrow R$' is correct, we should judge the validity of the applications of this rule: applications of a correct rule (within valid contexts for the rule) should generate correct inferences. On the other hand, applications of an incorrect rule generate

incorrect inferences. For example, applying the correct rule '*X* compose *Y* ⇒ *X* write *Y*' to '*Mozart composed over 600 works*' generates the correct inference '*Mozart wrote over 600 works*'. In contrast, applying the incorrect rule '*X* read *Y* ⇒ *X* write *Y*' to '*He reads the comics daily*' generates the incorrect inference '*He writes the comics daily*'.

Following, at the heart of the instance-based evaluation approach, human judges are presented not only with a rule but rather with a few examples of the rule's applications. Instead of thinking up valid contexts for the rule, the judges just need to assess the rule's validity under the given context in each example.

To describe how a rule application is tested, we look at rule instantiations: given a rule '$L \rightarrow R$' that is matched in a sentence, the phrase constructed by instantiating the left template *L* with the matched arguments in the sentence is termed the *left phrase*. Similarity, the phrase constructed by instantiating the right template *R* with the matched arguments is termed the *right phrase*. For example, the left and right phrases generated for the application of '*X* compose *Y* ⇒ *X* write *Y*' to '*Mozart composed over 600 works during his lifetime*' are '*Mozart compose over 600 works*' and '*Mozart write over 600 works*', respectively (ignoring verb inflections). To evaluate a rule application to a sentence, the question that assesses whether entailment holds is:

> **Q_{re}**: *Is the right phrase entailed from the sentence?* A positive/negative answer corresponds to an 'Entailment holds/No entailment' judgment.

See examples 5–8 in Table 6 for cases where entailment does or does not hold.

While the above question provides the core entailment assessment, there are cases in which we do not expect entailment to hold between the sentence and the right phrase. One such case is when even the left phrase is not entailed from the sentence. This may occur due to incorrect matching, following incorrect sentence analysis, or due to approximate matching of the left template in the sentence, ignoring for example semantic aspects like negation, modality and conditionals. Such issues could result in retrieved sentences that match the left template but do not entail the left phrase (see examples 1 and 2 in Table 6). While inappropriate matches of the rule's left-hand side may happen and harm precision of a whole system, such errors should be accounted for a system's rule matching module rather than for the rules precision. Thus, these applications should be ignored when evaluating the rule's correctness. The following question assesses this situation:

> **Q_{le}**: *Is the left phrase entailed from the sentence?* A positive/negative answer corresponds to a 'Left entailed/not entailed' judgment.

There is another case where we do not expect the rule to yield correct inference, even if by itself it is a correct rule. This happens if the context of the sentences is invalid for applying the rule. This may occur, for example, due to a mismatch between the meaning of the predicate occurrence in the sentence and its meaning in the rule. This case is demonstrated in examples 3 and 4 in Table 6. While many systems still apply rules without verifying the context validity for the rules, context-sensitive rule application is an active field of research (Pantel *et al.* 2007; Erk and Padó 2008; Szpektor *et al.* 2008) and we are interested in measuring the expected performance of a rule-set for systems with and without context-sensitive application capabilities. To capture this distinction, which allows

Table 6. *Rule evaluation examples and their judgment*

	Rule	Sentence	Judgment
1	*X* seek *Y*→*X* disclose *Y* Left phrase	If he is arrested, **he** can immediately seek **bail**. **He** seek **bail**	Left not entailed
2	*X* clarify *Y*→*X* prepare *Y* Left phrase	**He** didn't clarify **his position on the subject**. **He** clarify **his position on the subject**	Left not entailed
3	*X* hit *Y*→*X* approach *Y* Right phrase	**Earthquakes** have hit **Lebanon** since '82. **Earthquakes** approach **Lebanon**	Unrelated context
4	*X* lose *Y*→*X* surrender *Y* Right phrase	**Bread** has recently lost **its subsidy**. **Bread** surrender **its subsidy**	Unrelated context
5	*X* regulate *Y*→*X* reform *Y* Left phrase Right phrase	**The SRA** regulates **the sale of sugar**. **The SRA** regulate **the sale of sugar** **The SRA** reform **the sale of sugar**	No entailment
6	*X* resign *Y*→*X* share *Y* Left phrase Right phrase	**Lopez** resigned **his post at VW** last week. **Lopez** resign **his post at VW** **Lopez** share **his post at VW**	No entailment
7	*X* set *Y*→*X* allow *Y* Left phrase Right phrase	**The committee** set **the following refunds**. **The committee** set **the following refunds** **The committee** allow **the following refunds**	Entailment holds
8	*X* stress *Y*→*X* state *Y* Left phrase Right phrase	**Ben Yahia** also stressed **the need for action**. **Ben Yahia** stress **the need for action** **Ben Yahia** state **the need for action**	Entailment holds

us to optionally ignore irrelevant (unrelated) contexts when assessing rule correctness, the judges are asked another question:

> $\mathbf{Q_{rc}}$: *Is the right phrase a likely phrase in English?* A positive/negative answer corresponds to a 'Related/Unrelated context' evaluation.

If the right phrase is not likely in English, then the given context was probably drawn from a scenario that is not consistent with the meaning of the predicate in the rule. This is inherently related to the problem of language polysemy, which we must take into account.

To conclude, for each example sentence for a rule, the judges are presented with the given sentence and the left and right phrases. They are then asked the following ordered sequence of questions, which determines rule validity in the given sentence: (1) $\mathbf{Q_{le}}$, (2) $\mathbf{Q_{rc}}$, and (3) $\mathbf{Q_{re}}$. If the answer to a certain question is negative, then we do not need to present the next questions to the judge: if the left phrase is not entailed then we ignore the sentence altogether; and if the context is unrelated then the right phrase cannot be entailed from the sentence and so the answer to $\mathbf{Q_{re}}$ is already known as negative.

We compute the precision of a rule as the percentage of examples for which entailment holds out of all 'relevant' examples. We can calculate the precision in two ways, as defined below, depending on whether we ignore unrelated contexts or not (obtaining lower precision if we do not). When actual systems answer an information need, such as a query or question, unrelated contexts are sometimes not encountered thanks to additional context which is present in the given input. In addition, context-matching methods, such as

presented by Pantel *et al.* (2007) and Szpektor *et al.* (2008), may be applied to avoid such erroneous rule applications. Thus, we define the two measures:

$$\textbf{Strict-Context Precision} : \frac{\#\textbf{Entailment holds}}{\#\textbf{Relevant context}}$$

$$\textbf{Loose-Context Precision} : \frac{\#\textbf{Entailment holds}}{\#\textbf{Left entailed}}$$

where # denotes the number of examples with the corresponding judgment.

Finally, a rule is considered to be correct only if its precision is at least 80%, which seems sensible for typical applied settings. This yields two alternative sets of correct rules, corresponding to the Strict-Context and Loose-Context Precision measures. Even though judges may disagree on specific examples for a rule, their judgments may still agree overall on the rule's correctness. We therefore expect the agreement level on rule correctness to be higher than the agreement on judgments for individual examples.

6.2 Experimental setting

We applied the instance-based evaluation methodology to evaluate the TEASE algorithm. For comparison, we applied the very same setting to the DIRT algorithm (Lin and Pantel 2001), whose output was available upon request from the authors.

For every given input template I, each algorithm provides a list of learned output templates $\{O_j\}_1^{n_I}$, where n_I is the number of output templates learned for I. Each output template is suggested as holding an entailment relation with the input template I, but the algorithms (both TEASE and DIRT) do not specify the entailment directions. Thus, each pair $\{I, O_j\}$ induces two candidate directional entailment rules: '$I \rightarrow O_j$' and '$O_j \rightarrow I$'.

6.2.1 Test set construction

The test set construction consists of three sampling steps needed in order to obtain a manageable-size test: selecting a set of input templates for the two algorithms, selecting a sample of output rules to be evaluated, and selecting a sample of rule applications (over a sample of sentences) to be judged for each rule.

First, we randomly selected 30 transitive verbs out of the 1,000 most frequent verbs in the Reuters RCV1 corpus.[8] For each verb, we manually constructed a lexical–syntactic input template by adding subject and object variables. For example, for the verb 'seek' we constructed the template '$X \xleftarrow{subj}$ seek $\xrightarrow{obj} Y$'.

Next, for each input template I we considered the learned templates $\{O_j\}_1^{n_I}$ from each algorithm's output. We then sampled 10% of the templates in each output list, limiting the sample size to between 5 and 20 templates for each list (thus balancing between sufficient evaluation data and judgment load). For each sampled template O we evaluated both directional rules, '$I \rightarrow O$' and '$O \rightarrow I$'. In total, we sampled 209 templates out of the TEASE output, inducing 418 directional rules.

[8] Available at http://about.reuters.com/researchandstandards/corpus/

Last, we randomly extracted a sample of example sentences for each rule '$L \rightarrow R$' by utilizing a search engine over the first CD of Reuters RCV1. First, we retrieved all sentences containing all lexical terms within L. The retrieved sentences were parsed using Minipar, keeping only sentences that syntactically match L. A sample of 15 matching sentences was randomly selected, or all matching sentences were taken if fewer than 15 were found. Finally, an example for judgment was obtained from each sampled sentence along with its generated left and right phrases, as described in Section 6.1. For example, the left and right phrases generated for example 8 in Table 6 are '*Ben Yahia stress the need for action*' and '*Ben Yahia state the need for action*', respectively. For TEASE, we did not find sentences for 72 rules and, thus, we ended up with 346 unique rules that could be evaluated (with 4,740 TEASE examples to be judged).

The same scheme was also applied to DIRT over the same 30 transitive verbs. Since DIRT's output has a long tail of templates with a low score and very low precision, when sampling the 10% of templates in DIRT's output lists, the templates whose score is below a threshold of 0.1 were filtered out.[9] For DIRT, there were 171 templates making 342 directional rules, 340 of which were unique. When matching sentences, 38 rules resulted in no match, so 304 rules could be evaluated, eventually yielding 4295 associated sentences.

6.2.2 *Assessment of the instance-based evaluation methodology*

Two human judges evaluated the examples collected in the above setting. We randomly split the examples between the judges. One hundred rules (*i.e.* 1,287 examples overall, from TEASE and DIRT together) were cross annotated for agreement measurement. The judges followed the procedure in Section 6.1[10] and the correctness of each rule was assessed based on both its Strict-Context and Loose-Context Precision values.

We assessed the instance-based evaluation methodology by measuring the agreement level between the judges. The κ values for the final correctness judgments of the shared rules were 0.74 and 0.68 for the strict and loose evaluations. These κ scores are regarded as 'substantial agreement' and are substantially higher than published agreement scores and those we managed to obtain using the standard rule-based evaluation approach discussed in Section 6.

Table 7 illustrates some disagreements that were still exhibited within the instance-based evaluation. The primary reason for disagreements was the difficulty to decide whether a context is relevant for a rule or not, resulting in some confusion between 'Unrelated context' and 'No entailment'. This may explain the lower agreement for the Strict-Context Precision, for which examples judged as 'Unrelated context' are ignored, while for the Loose-Context both judgments are conflated and represent no entailment. Our findings suggest that better ways for distinguishing related contexts may be sought in future research for further refinement of the instance-based evaluation methodology.

[9] The threshold was set following the advice of DIRT's authors.

[10] The complete guidelines to the judges can be viewed at: https://sites.google.com/site/idanszpektor/ACL07_evaluation_guidelines.doc?attredirects=0&d=1

Table 7. *Examples for disagreement between the two judges*

Rule	Sentence	Judge 1	Judge 2
X sign *Y*→*X* set *Y*	**Iraq and Turkey** sign **agreement** to increase trade cooperation	Entailment holds	Unrelated context
X worsen *Y*→*X* slow *Y*	**News on the strike** worsened **the situation**	Unrelated context	No entailment
X get *Y*→*X* want *Y*	**He** will get **his parade** on Tuesday	Entailment holds	No entailment

7 Results and use case

An entailment engine would aim to utilize all its available resources for maximum coverage. These include rules learned from local corpora and rules from manually constructed resources, such as WordNet (Bar-Haim *et al.* 2006; Giampiccolo *et al.* 2007; Giampiccolo *et al.* 2008). Yet, until now, the Web was not utilized as a source for generating lexical–syntactic entailment rules. In this evaluation, we want to show that TEASE may be added to the pool of resources utilized by an entailment engine. Accordingly, our evaluation of the TEASE algorithm has two goals. First, we want to show TEASE's added value, which is the capability of learning rules that can be extracted neither from a local corpus nor from manually constructed resources. Our second goal is to show that TEASE's quality is not falling behind other unsupervised algorithms currently in use, in order to ensure that adding TEASE output as part of the pool of available entailment-rule resources would increase coverage without significantly hurting precision.

To this end, we compared TEASE with DIRT, the most widely used algorithm for learning rules from a local corpus, and to WordNet, the prominent manual resource for rules. We first compared TEASE and DIRT based on the manual evaluation setting described in Section 6. We then show the results of applying TEASE (and DIRT) output to an RE use case, in which the benefits of the approach taken by TEASE are further discussed. To complete this evaluation, we provide detailed error analysis for the rules learned by TEASE. Finally, we also compare TEASE's output to entailment rules that can be extracted from WordNet, to better understand the added value in TEASE's Web-based rule-set compared to manual resources.

7.1 DIRT comparison

We evaluated the quality of the entailment rules produced by TEASE and DIRT using two scores: (1) micro-averaged Precision, the percentage of correct rules out of all learned rules, and (2) average Yield, the average number of correct rules learned for each input template I, as extrapolated based on the evaluated sample. We remind the reader that since both DIRT and TEASE do not identify rule directionality, two distinct directional rules are evaluated for each output template O, i.e. the rule '$I \rightarrow O$' and the rule '$O \rightarrow I$' (see Section 6.2.1). In addition, we measured the Precision and Yield scores at the template

Table 8. *Average Precision (P) and Yield (Y) at the rule and template levels*

	TEASE		DIRT	
	P	Y	P	Y
Rules				
Strict-Context	28.4%	40.3	30.5%	33.5
Loose-Context	17%	24.1	18.6%	20.4
Templates				
Strict-Context	38%	26.9	44%	22.6
Loose-Context	23.6%	16.8	27.3%	14.1

level, where an output template O is considered correct if at least one of the rules '$I \rightarrow O$' or '$O \rightarrow I$' is correct. Our results are presented in Table 8.

Addressing our second evaluation goal, the figures in Table 8 indicate that the quality of DIRT and TEASE is very similar. Under the specific DIRT cutoff threshold chosen, DIRT exhibits somewhat higher Precision while TEASE has somewhat higher Yield (recall that there is no particular natural cutoff point for DIRT's output). This means that adding TEASE rules along side DIRT rules should not hurt performance.

Yet, our major finding from the evaluation annotations, addressing our first evaluation goal, is the low overlap between TEASE and DIRT: only about 15% of the correct templates were learned by both algorithms. This shows TEASE's added value, as this result implies that the two algorithms largely complement each other in terms of coverage and should be used in tandem. One explanation may be that DIRT is focused on the domain of the local corpus used (news articles for the published DIRT knowledgebase), whereas TEASE learns from the Web, extracting rules from multiple domains. We further demonstrate this difference in our application use-case (Section 7.2), showing the applicability of TEASE to open-domain IE, in which the target domain is not known in advance.

Since applications typically apply rules in a specific direction, the Precision for rules reflects their expected performance better than the Precision for templates. Obviously, future improvement in precision is needed for unsupervised rule learning algorithms. Meanwhile, manual filtering of the learned rules can prove effective within limited domains, where our evaluation approach can be utilized for reliable filtering as well. The substantial yield obtained by these algorithms suggests that they are indeed likely to be valuable for recall increase in semantic applications.

We also measured whether O is a paraphrase of I, *i.e.* whether both '$I \rightarrow O$' and '$O \rightarrow I$' are correct. Only 20%–25% of all correct output templates were assessed as paraphrases. This stresses the significance of evaluating directional rules rather than only symmetric paraphrases, as was originally done for DIRT. Furthermore, it shows that in order to improve precision, acquisition algorithms must identify rules directionality.

Finally, about 28% of all 'Left entailed' examples were evaluated as 'Irrelevant context', yielding the large difference in precision between the Strict-Context and Loose-Context measures. This result shows that in order to get closer to the strict precision, learning

algorithms and applications need to identify the relevant contexts in which a rule should be applied.

7.2 Use case: relation extraction

Our main evaluation setup compared TEASE and DIRT by assessing the relations learned for input templates that come from the domain of the DIRT knowledgebase we utilized, namely the news domain. To further show the added value of TEASE, providing additional coverage over DIRT and other local corpora rule-sets, we present in this subsection an analysis of an experiment of applying TEASE to an open IE setting.

To this end, we summarize an experiment with TEASE for the task of RE on a data set containing textual descriptions of protein interactions. This experiment, originally reported in Romano *et al.* (2006), aims at extracting mentions of protein–protein interactions from a given corpus with minimal supervision, using Textual Entailment as the relation detection framework and TEASE rules providing the entailment knowledge. Textual Entailment is highly suitable for RE since its goal is exactly to identify the different variations in which a target semantic relation can be expressed. Before describing the experiment, we would like to note two interesting features of the task at hand. First, it is situated in a different domain than the news domain and thus shows how DIRT and TEASE would perform on a 'domain transfer' task. Second, in our primary manual evaluation, no notion of absolute recall could be provided, only the number of correct rules extracted. In the IE setting, however, all protein interactions are annotated in a given test corpus, so recall can be measured. We thus focus our analysis on the coverage performance of TEASE and DIRT for the task.

In order to detect mentions of protein–protein interactions, Romano *et al.* learned templates that have an entailment relation to the input template '*X* interact with *Y*', which was the only manual input provided to the system. Here, all learned templates were considered as entailing the input template. The top 18 correct templates learned are shown in Table 9. It is interesting to note that TEASE learned relevant templates for the protein interaction domain also by finding anchor-sets in different domains, which use the same jargon, for example referring to particle instantiations in physics. This shows a benefit of using the Web, which covers many different domains, when a larger rule coverage for a target domain can be achieved via related domains.

To measure the potential Recall of TEASE on this task, assuming perfect template matching, Romano *et al.* manually inspected all protein interaction mentions that are annotated in the test data set. They identified all such mentions in which the interaction is expressed by an entailing template which was learned by TEASE (see Romano *et al*. 2006

Table 9. *The top 18 correct templates learned by TEASE for* 'X *interact with* Y'

1. *X bind to Y*	7. *X Y complex*	13. *X interaction with Y*
2. *X activate Y*	8. *X recognize Y*	14. *X trap Y*
3. *X stimulate Y*	9. *X block Y*	15. *X recruit Y*
4. *X couple to Y*	10. *X binding to Y*	16. *X associate with Y*
5. *Interaction between X and Y*	11. *X Y interaction*	17. *X be linked to Y*
6. *X become trapped in Y*	12. *X attach to Y*	18. *X target Y*

Table 10. *The potential Recall of TEASE in terms of distinct pairs of interacting proteins (out of 418 annotated pairs)*

Experiment	Pairs (%)
Input	39
Input+Iterative	49
Input+Iterative+Morph	63

for details). Both the number of mentions and the number of distinct interacting protein pairs that are covered by the TEASE output were counted. This process results in an upper bound for the Recall of a TEASE-based RE approach, reflecting the coverage of the relation's mentions by the TEASE output. In practice, somewhat lower recall is expected for an actual due to imperfect syntactic template matching.

Table 10 presents the results of this analysis. The first line shows that utilizing the templates learned by TEASE for the input template '*X* interact with *Y*', a 39% Recall can be reached. The second line shows the contribution of *bootstrapping* to Recall. Bootstrapping (see Section 3.2) was performed by taking the top five output templates learned by TEASE as input, and then learning additional entailment templates for them, which should entail the original input template as well via transitivity. With the additional learned templates, Recall increased by 10%. The final line in the table shows the contribution of improved template matching, which identifies also morphological variations of each template, such as nominalizations, in addition to syntactic variations.[11] Assuming the usage of such a matcher increases TEASE recall by additional 14%, reaching 63%. Thus, the potential recall of TEASE on the protein interaction data set is estimated to be above 60%. This indicates that significant coverage can be obtained using completely unsupervised learning from the Web, as performed by TEASE. To demonstrate the extraction system Romano *et al.* utilized a basic matcher that reached 18% Recall for the input template and 29% Recall with bootstrapping. The precision for these two rule-sets was 62% and 42%, respectively, showing that TEASE precision for this task is above average (28%, Table 8) even with bootstrapping.

To complete this analysis, we extracted rules for '*X* interact with *Y*' using the available DIRT database. The resulting rules for the same protein interaction task yielded no correct extractions (protein interactions) at all. The reason is that the DIRT database was learned from a local corpus constructed from news articles. Thus, no rules that were useful for the tested biomedical domain were learned. A local corpus for protein interaction could have been utilized by the DIRT algorithm, but the effort spent for constructing and utilizing such a corpus would have been much longer than the execution of TEASE for the target relation over the Web. This indicates the added value of using the Web as a large heterogeneous corpus, which is readily utilized for different domains in an open IE setup, whereas DIRT relies on a specific corpus to be collected for each task. In addition, we also extracted rules for '*X* interact with *Y*' from WordNet (see rule generation description in Section 7.4),

[11] Examples for such existing matchers are described in Bar-haim *et al*. (2007) and Szpektor and Dagan (2007).

Table 11. *Distribution of reasons for learning incorrect templates by TEASE*

Reason	Percentage
Non-indicative anchor-set	62.0
Too-specific template	17.4
Partial template	1.0
Parse error	8.7
Annotation disagreement	10.9

which resulted only in one useful rule for the task, '*X* associate with *Y*'. This again shows the benefit of a Web-based heterogeneous rule-set, since domain specific senses of 'interact with', as well as their relationships, are not currently encoded in WordNet.

7.3 Error analysis

We next turn to analyze the reasons for learning incorrect templates by TEASE. To this end, we analyzed the TEASE templates found as incorrect by out annotators. We partitioned the reasons for such templates into five main categories: *non-characteristic anchor set*, *too specific template*, *partial template*, *parse error*, *annotation disagreement*. The distribution of errors for these relations is shown in Table 11. We next discuss each of these categories.

Non-characteristic anchor -set. Many erroneous templates are extracted due to anchor-sets that are not characteristic, and thus occur with templates that are semantically related but not necessarily in an entailment relation with the input template. Such examples are ⟨*X* stand *Y*, *X* sit *Y*⟩ via {*X*='*elderly*', *Y*='*upright*'} and ⟨*X* want *Y*, *X* get *Y*⟩ via {*X*='*dog*', *Y*='*cracker*'}. In more acute cases, some of the anchor-sets are simply incorrect due to parse errors, resulting in completely unrelated templates, *e.g.* ⟨*X* calculate *Y*, *X* number *Y*⟩ via the anchor-set {*X*='*isotherms*', *Y*='*equation*'}.

Too-specific template. These templates are the result of a long complex sentence in which the matched anchors are placed at distant parts of the parse trees. Thus, the resulting template that includes them is too long and complex to be useful in a general case. For example, '*X* worsen and destroy *Y*' was learned for the input '*X* worsen *Y*'. While the specific template may entail the input if in its rare occurrences, it is too specific to be considered as correctly entailed by the input.

Partial template. These templates are the result of the definition of the LGMG templates that are learned by the TEASE algorithm (Section 5.2.1): each node and edge in the extracted template must appear in all occurrences of the path that this LGMG template generalizes. While improving over algorithms like DIRT, which look only for paths between arguments, this constraint is sometimes too strict, preventing from learning the correct templates. For example, the templates '*X* convince *Y* to quit' and '*X* convince *Y* to stop', which should have been learned for '*X* stop *Y*', are not extracted. Instead, the partial path template '*X* convince *Y*' is learned, because it occurs in more sentences than '*X* convince *Y* to quit' or '*X* convince *Y* to stop', and thus cannot be extended to them via

the GSL algorithm. Future work may investigate a softer template extension algorithm to address this limitation. This is especially important due to the significant harmful effect of partial templates when utilizing a rule-set, since partial templates occur in texts much more than complete templates (Szpektor and Dagan 2008).

Parser errors. Whenever the parser incorrectly parses a sentence, this erroneous information may indirectly affect all of the above categories, for example extracting an incorrect anchor set, or producing a partial template. However, in this category we focus only on templates that are unrelated to the input template because of the wrong tree structure. For example, the template '*X* involve *Y*' was learned for '*X* seek *Y*' due to several parse errors that placed the anchor-sets in incorrect positions in the sentence tree. In a more general note, many pages on the Web may add more noise than valid content. Therefore, some pre-filtering, such as removing snippets with bad parse trees, may significantly improve the extraction quality of the TEASE algorithm.

Annotation disagreement. While the annotation task improved with the instance-based methodology we used, annotators still make mistakes, and we found that to be about 10% of the cases. These templates are usually correct only for rare contexts, in which the typical contexts found in the corpus, and thus the annotators miss the correct contexts under which the rules are correct. This is, of course, expected from this type of annotation, as stated in Szpektor *et al.* (2007), where the selected corpus for examples indicates the types of contexts for under which the rules should be judged, assuming other contexts are not as interesting. Some examples for such correct relations are $\langle X \text{ set } Y, X \text{ change } Y \rangle$ and $\langle X \text{ seek } Y, X \text{ call for } Y \rangle$.

7.4 WordNet comparison

To further analyze the added value of our approach in terms of coverage, we compared TEASE's output with the lexical–syntactic entailment rules that can be extracted from WordNet[12] (Miller 1995). To this end, we derived from WordNet templates that are in entailment relation with our input test templates as follows:

1. The lexical element of an input template is extracted. For example, 'acquire' is extracted as the lexical element of '$X \xleftarrow{subj}$ acquire $\xrightarrow{obj} Y$'.
2. Lexical entailment relations are acquired from WordNet. For example, $\langle acquire, buy \rangle$.
3. Templates are generated by replacing the lexical element in the input template with the terms acquired in the previous step. For example, the entailment relation $\langle X \xleftarrow{subj}$ acquire $\xrightarrow{obj} Y, X \xleftarrow{subj}$ buy $\xrightarrow{obj} Y \rangle$ is generated.

Since WordNet does not specify the syntactic changes involved in applying the lexical rules extracted from it, we performed a basic substitution between the two terms in step 3. Thus, in step 2 we restricted ourselves to learning *substitutable* lexical rules, whose terms

[12] We used WordNet 3.0.

Table 12. *Correct templates that were learned by TEASE and cannot be extracted from WordNet*

Input template	Correct templates by TEASE not in WordNet
$X \xleftarrow{subj}$ demand $\xrightarrow{obj} Y$	$X \xleftarrow{subj}$ seek $\xrightarrow{obj} Y$, $X \xleftarrow{subj}$ want $\xrightarrow{obj} Y$
$X \xleftarrow{subj}$ leave $\xrightarrow{obj} Y$	$X \xleftarrow{subj}$ enter $\xrightarrow{obj} Y$, $X \xleftarrow{subj}$ return to $\xrightarrow{obj} Y$
$X \xleftarrow{subj}$ write $\xrightarrow{obj} Y$	$X \xleftarrow{subj}$ produce $\xrightarrow{obj} Y$, $X \xleftarrow{subj}$ complete $\xrightarrow{obj} Y$
$X \xleftarrow{subj}$ establish $\xrightarrow{obj} Y$	$X \xleftarrow{subj}$ develop $\xrightarrow{obj} Y$, $X \xleftarrow{subj}$ propose $\xrightarrow{obj} Y$
$X \xleftarrow{subj}$ tell $\xrightarrow{obj} Y$	$X \xleftarrow{subj}$ warn $\xrightarrow{obj} Y$, $X \xleftarrow{subj}$ teach $\xrightarrow{obj} Y$
$X \xleftarrow{subj}$ acquire $\xrightarrow{obj} Y$	$X \xleftarrow{subj}$ eye $\xrightarrow{obj} Y$, $Y \xleftarrow{subj}$ subsidiary $\xrightarrow{of} X$
$X \xleftarrow{subj}$ name $\xrightarrow{as} Y$	$X \xleftarrow{subj}$ appoint $\xrightarrow{obj} Y$, $X \xleftarrow{subj}$ make $\xrightarrow{desc} Y$
$X \xleftarrow{subj}$ kill $\xrightarrow{obj} Y$	$X \xleftarrow{subj}$ claim $\xrightarrow{obj}$ life $\xrightarrow{of} Y$, $Y \xleftarrow{subj}$ die $\xrightarrow{in} X$

can indeed be substituted one for the other in a sentence without changing its meaning and without the need to further change its syntactic structure (with respect to argument positions). To this end, we considered the *hypernymy* semantic relation between verb synsets, *e.g.* 'buy back ⇒ buy'. On top of that, all terms in each synset entail each other. Since we do not care for the direction of entailment, we generated both entailing relations and entailed relations. In addition, we generated relations based on all possible chains of the above two lexical relations.

We measured the percentage of lexical–syntactic relations extracted by TEASE that also appear in the lexical–syntactic relations extracted from WordNet. We found that on average, about 60% of the correct templates extracted by TEASE cannot be extracted by WordNet. This figure emphasizes the added coverage obtained by using corpus-based extraction methods such as TEASE over using WordNet alone (and likely other manually constructed resources). Table 12 presents examples for rules learned by TEASE that were not acquired from WordNet. Some of these examples include templates with substitutable relations between the lexical elements of the two templates, showing that WordNet is lacking in directional entailment relations, such as $\langle warn, tell \rangle$, especially regarding informal language or slang, *e.g.* $\langle eye, acquire \rangle$. However, there are also examples that involve syntactic changes to argument positions on top of the lexical change, *e.g.* $\langle X \xleftarrow{subj}$ kill $\xrightarrow{obj} Y$, $Y \xleftarrow{subj}$ die $\xrightarrow{in} X \rangle$. Such information is currently not part of WordNet's database.

Following our last example, as a final analysis we took into consideration also non-substitutable entailment relations in WordNet, such as *cause*, even though WordNet cannot turn them into correct lexical–syntactic rule (*e.g.* for ⟨kill, die⟩ above). To this end, we compared only the lexical part of the rules learned by TEASE and by WordNet and found that about 53% of the correct TEASE rules cannot be extracted from WordNet. This indicates that non-substitutable relations, which may involve complex syntactic changes, are still rather scarce in WordNet. We note that 53% is a lower bound, since some of the TEASE rules that are considered incorrect in this experiment may be so only due to incorrect argument mapping, *i.e.* incorrectly learning the lexical–syntactic entailment

relation $\langle X \xleftarrow{subj} \text{kill} \xrightarrow{obj} Y, X \xleftarrow{subj} \text{die} \xrightarrow{in} Y \rangle$), even though the entailment relation between the lexical parts of the two templates is correct.

8 Conclusions and discussion

This article presented TEASE, the first fully unsupervised algorithm for the acquisition of entailment relations from the Web. The algorithm provides several improvements over prior work. First, our unsupervised approach for acquiring characteristic anchor-sets overcomes the main scalability limitation of prior Web-based work, in which manual input of seed anchor-sets is required for each input template. Second, in the template-extraction phase, the GSL algorithm enables to efficiently learn entailment relations between arbitrary lexical–syntactic templates, without any *a priori* restriction on template structure. Rather, the template structure is learned in an unsupervised way from the data, which is used to discover the entailment relations.

Our experiments show that the quality of TEASE rules is comparable to those of DIRT, the most commonly used large-scale rule base. However, the rules learned by the two algorithms are complementary, as DIRT extracts rules from a local corpus, typically of a specific domain, while TEASE learns rules for many different senses of a template from the heterogeneous content of the Web. We thus conclude that it is best to use these two algorithms in tandem. In addition, we showed that TEASE learns many entailment relations that cannot be found in WordNet. Finally, in use case experiment of extracting mentions of protein interactions, we demonstrated the potential of TEASE to learn many of the entailment rules needed to cover the different ways of expressing this target relation.

Our work suggests interesting directions for future research. First, while the focus of this article is to increase the scalability of Web-based rule acquisition, a following step would be to further improve the quality of the acquired rules. As shown in prior work, incorporating additional features for filtering entailment between predicates, on top of distributional similarity, could improve the quality of the filtered rule-set. This is typically done under a supervised framework (Berant, Dagan and Goldberger 2012; Weisman *et al.* 2012). Similarly, the accuracy of characteristic anchor-sets could be further improved using supervised learning, *i.e.* learning a classifier for detecting the characteristic attributes of anchor-sets. For concrete action predicates, *e.g.* '*acquire*', '*celebrate*' and '*write*', this is less of a problem since dozens of characteristic anchor-sets can be detected with our algorithm. But for predicates such as '*want*' or '*think*' this is important due to the relative scarcity of characteristic anchor-sets. Last, using typed templates (Schoenmackers *et al.*2010; Nakashole, Weikum and Suchanek 2012) may additionally improve the accuracy of the extracted rules.

Second, the rule structure may be improved. One research line could be identifying the direction of the entailment relations learned by TEASE, using directional measures such as in Chklovski and Pantel (2004), Bhagat *et al.* (2007), Szpektor and Dagan (2008) and Kotlerman *et al.* (2010). In addition, in this work we propose one way to efficiently learn more general template structures than could be obtained in most prior work, namely a dependency path. It would be interesting to explore and compare other complementary approaches for generalizing and extending the template structure, include lexical

generalization (Mausam *et al.* 2012) and dependency paths for single-argument templates (Szpektor and Dagan 2008).

Finally, the TEASE algorithm was designed to harness the Web as a textual resource. However, the methodologies used in the algorithm are not limited for the Web. We propose to adapt TEASE to process local corpora, in order to test whether the methodologies we developed for the Web are similarly effective for such corpora.

Acknowledgments

This work was partially supported by ISF grant 1095/05, the IST Programme of the European Community under the PASCAL Network of Excellence IST-2002-506778, and the ITC-irst/University of Haifa collaboration.

References

Androutsopoulos, I., and Malakasiotis, P. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research* **38**: 135–87.

Baeza-Yates, R., and Raghavan, P. 2010. Chapter 2: next generation web search. In S. Ceri, and M. Brambilla (eds.), *Search Computing*, pp. 11–23. Lecture Notes in Computer Science, vol. 5950. Berlin/Heidelberg: Springer.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics,* Vol. 1 *(ACL '98)*, pp. 86–90. Stroudsburg, PA: Association for Computational Linguistics.

Bannard, C. J., and Callison-burch, C. 2005. Paraphrasing with bilingual parallel corpora. In *Meeting of the Association for Computational Linguistics*, Ann Arbor, Michigan.

Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. 2006. The second PASCAL recognising textual entailment challenge. In *Second PASCAL Challenge Workshop for Recognizing Textual Entailment*, Venice, Italy.

Bar-haim, R., Dagan, I., Greental, I., and Shnarch, E. 2007. Semantic inference at the lexical-syntactic level. In *National Conference on Artificial Intelligence*, Vancouver, British Columbia, Canada.

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

Barzilay, R., and Lee, L. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *North American Chapter of the Association for Computational Linguistics*, vol. cs.CL/0304, Stroudsburg, PA, USA.

Barzilay, R., and McKeown, K. R. 2001. Extracting paraphrases from a parallel corpus. In *Meeting of the Association for Computational Linguistics*, Toulose, France, pp. 50–57.

Ben Aharon, R., Szpektor, I., and Dagan, I. 2010. Generating entailment rules from FrameNet. In *Proceedings of the ACL 2010 Conference Short Papers*, pp. 241–6. Uppsala, Sweden: Association for Computational Linguistics.

Bentivogli, L., Dagan, I., Dang, H. T., Giampiccolo, D., and Magnini, B. 2009. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the TAC 2009 Workshop*, Gaithersburg, Maryland, USA.

Bentivogli, L., Clark, P., Dagan, I., Dang, H. T., and Giampiccolo 2010. The sixth PASCAL recognizing textual entailment challenge. In *Proceedings of the TAC 2010 Workshop*, Gaithersburg, Maryland, USA.

Berant, J., Dagan, I., and Goldberger, J. 2012. Learning entailment relations by global graph structure optimization. *Computational Linguistics* **38**(1): 73–111.

Bergadano, F., and Gunetti, D. 1995. *Inductive Logic Programming: From Machine Learning to Software Engineering*. Cambridge, MA: MIT Press.

Bhagat, R., Pantel, P., and Hovy, E. H. 2007. LEDIR: an unsupervised algorithm for learning directionality of inference rules. In *Empirical Methods in Natural Language Processing*, Prague, Czech Republic, pp. 161–70.

Carletta, J. 1996. Assessing agreement on classification tasks: the Kappa statistic. *Computational Linguistics* **22**(2): 249–54.

Chklovski, T., and Pantel, P. 2004. VerbOcean: mining the web for fine-grained semantic verb relations. In D. Lin and D. Wu (eds.), *Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Barcelona, Spain, pp. 33–40.

Condoravdi, C., Crouch, D., de Paiva, V., Stolle, R., and Bobrow, D. G. 2003. Entailment, intensionality and text understanding. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, Stroudsburg, PA, USA.

Dagan, I., and Glickman, O. 2004. Probabilistic textual entailment: generic applied modeling of language variability. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*, Grenoble, France.

Dagan, I., Glickman, O., and Magnini, B. 2006. The PASCAL recognising textual entailment challenge. In J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché-Buc (eds.), *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment,* Lecture Notes in Computer Science, Vol. 3944, pp. 177–90. Berlin: Springer.

Dolan, B., Quirk, C., and Brockett, C. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *International Conference on Computational Linguistics*, Stroudsburg, PA, USA.

Duclaye, F., Yvon, F., and Collin, O. 2002. Using the web as a linguistic resource for learning reformulations automatically. In *Language Resources and Evaluation*, Las Palmas, Spain, pp. 390–96.

Durme, B. Van, Huang, Y., Jupść, A., and Nyberg, E. 2003. Towards light semantic processing for question answering. In *Proceedings of HLT/NAACL Workshop on Text Meaning 2003*, Stroudsburg, PA, USA.

Erk, K., and Padó, S. 2008. A structured vector space model for word meaning in context. In *Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA, pp. 897–906.

Giampiccolo, D., Dang, H. T., Magnini, B., Dagan, I., Cabrio, E., and Dolan, B. 2008. The forth PASCAL recognizing textual entailment challenge. In *Proceedings of the TAC 2008 Workshop*, Gaithersburg, Maryland, USA.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Prague, Czech Republic.

Glickman, O., and Dagan, I. 2003. Identifying lexical paraphrases from a single corpus: a case study for verbs. In *Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria.

Harabagiu, S., and Hickl, A. 2006. Methods for using textual entailment in open-domain question answering. In *Meeting of the Association for Computational Linguistics*, Sydney, Australia, pp. 905–12.

Hermjakob, U., Echihabi, A., and Marcu, D. 2003. Natural language based reformulation resource and web exploitation. In E. M. Voorhees and L. P. Buckland (eds.), *Proceedings of the 11th Text Retrieval Conference (TREC 2002)*. Gaithersburg, MD: NIST.

Ibrahim, A., Katz, B., and Lin, J. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the Second International Workshop on Paraphrasing (IWP-2003)*, Sapporo, Japan.

Jacquemin, C. 1999. Syntagmatic and paradigmatic representations of term variation. In *Meeting of the Association for Computational Linguistics*, College Park, Maryland, USA.

Kietz, J.-U., and Lubbe, M. 1994. An efficient subsumption algorithm for inductive logic programming. In *ICML*, New Brunswick, NJ, USA.

Kipper, K., Dang, H. T., and Palmer, M. S. 2000. Class-based construction of a verb lexicon. In *National Conference on Artificial Intelligence*, Austin, Texas, USA, pp. 691–6.

Kotlerman, L., Dagan, I., Szpektor, I., and Zhitomirsky-Geffet, M. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* **16**(4): 359–89.

Lin, D. 1998. Dependency-based evaluation of Minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC 1998*, Granada, Spain.

Lin, D., and Pantel, P. 2001. Discovery of inference rules for question answering. *Natural Language Engineering* **7**(4): 343–60.

Lloret, E., Ferrández, Ó., Muñoz, R., and Palomar, M. 2008. A text summarization approach under the influence of textual entailment. In *Natural Language Understanding and Cognitive Science*, Barcelona, Spain, pp. 22–31.

Markov, Z., and Pelov, N. 1998. A framework for inductive learning based on subsumption lattices. In F. Giunchiglia (ed.), *Artificial Intelligence: Methodology, Systems, and Applications, Proceedings of the 8th International Conference (AIMSA 98)*, pp. 341–52. Lecture Notes in Computer Science, vol. 1480. Sozopol, Bulgaria: Springer.

Mausam, M. S., Bart, R., Soderland, S., and Etzioni, O. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '12)*, pp. 523–34. Stroudsburg, PA: Association for Computational Linguistics.

Miller, G. A. 1995. WordNet: a lexical database for English. *Communications of The ACM* **38**: 39–41.

Mirkin, S., Dagan, I., and Shnarch, E. 2009. Evaluating the inferential utility of lexical–semantic resources. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 558–566. Athens, Greece: Association for Computational Linguistics.

Moldovan, D., and Rus, V. 2001. Logic form transformation of WordNet and its applicability to Question Answering. In *Proceedings of ACL 2001*, Toulose, France, pp 394–401.

Monz, C., and de Rijke, M. 2001. Light-weight entailment checking for computational semantics. In *Proceedings of the Third Workshop on Inference in Computational Semantics (ICoS-3)*, Italy.

Nakashole, N., Weikum, G., and Suchanek, F. 2012. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP–CoNLL'12)*, pp. 1135–45. Stroudsburg, PA: Association for Computational Linguistics.

Pang, B., Knight, K., and Marcu, D. 2003. Syntax-based alignment of multiple translations: extracting paraphrases and generating new sentences. In *North American Chapter of the Association for Computational Linguistics*, Edmonton, Canada.

Pantel, P., Bhagat, R., Coppola, B., Chklovski, T., and Hovy, E. 2007. ISP: learning inferential selectional preferences. In *Proceedings of North American Association for Computational Linguistics/Human Language Technology Conference (NAACL HLT 07)*, Rochester, New York.

Pantel, P., and Pennacchiotti, M. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pp. 113–20. Stroudsburg, PA: Association for Computational Linguistics.

Pekar, V. 2006 (June). Acquisition of verb entailment from text. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA, pp. 49–56.

Ravichandran, D., and Hovy, E. H. 2002. Learning surface text patterns for a Question Answering system. In *Meeting of the Association for Computational Linguistics*, Philadelphia, PA, pp. 41–7.

Reynar, J. C., and Ratnaparkhi, A. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the fifth conference on Applied natural language processing*, pp. 16–9. Stroudsburg, PA: Association for Computational Linguistics.

Reynolds, J. C. 1970. Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence* **5**: 135–51.

Romano, L., Kouylekov, M., Szpektor, I., Dagan, I., and Lavelli, A. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy.

Schoenmackers, S., Etzioni, O., Weld, D. S., and Davis, J. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*, pp. 1088–98. Stroudsburg, PA: Association for Computational Linguistics.

Sekine, S. 2005. Automatic paraphrase discovery based on context and keywords between NE pairs. In *The 3rd International Workshop on Paraphrasing*, Jeju Island, South Korea.

Shinyama, Y., and Sekine, S. 2006. Preemptive information extraction using unrestricted relation discovery. In *North American Chapter of the Association for Computational Linguistics*, New York City, USA.

Shinyama, Y., Sekine, S., Sudo, K., and Grishman, R. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of Human Language Technology Conference (HLT 2002)*, San Diego, CA, USA.

Suchanek, F. M., Ifrim, G., and Weikum, G.. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Knowledge Discovery and Data Mining*, Philadelphia, USA, pp. 712–7.

Sudo, K., Sekine, S., and Grishman, R. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Meeting of the Association for Computational Linguistics*, Sapporo, Japan, pp. 224–31.

Szpektor, I., and Dagan, I. 2007. Learning canonical forms of entailment rules. In *Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria.

Szpektor, I., and Dagan, I. 2008. Learning entailment rules for unary templates. In *International Conference on Computational Linguistics*, Manchester, UK, pp. 849–56.

Szpektor, I., Dagan, I., Bar-Haim, R., and Goldberger, J. 2008 (June). Contextual preferences. In *Proceedings of ACL-08: HLT*, pp. 683–91. Columbus, OH: Association for Computational Linguistics.

Szpektor, I., Shnarch, E., and Dagan, I. 2007. Instance-based evaluation of entailment rule acquisition. In *Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

Szpektor, I., Tanev, H., Dagan, I., and Coppola, B. 2004. Scaling web based acquisition of entailment patterns. In *Empirical Methods in Natural Language Processing*, Barcelona, Spain, pp. 41–8.

Tanev, H. 2007. Unsupervised learning of social networks from a multiple-source news corpus. In *Proceedings of the Workshop Multi-source Multilingual Information Extraction held at RANLP 2007*, Borovets, Bulgaria.

Tanev, H., and Magnini, B. 2006. Weakly supervised approaches for ontology population. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Weisman, H., Berant, J., Szpektor, I., and Dagan, I. 2012. Learning verb inference rules from linguistically-motivated evidence. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, (EMNLP-CoNLL '12)*, pp. 194–204. Stroudsburg, PA: Association for Computational Linguistics.

Zhao, S., Wang, H., Liu, T., and Li, S. 2008. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *Meeting of the Association for Computational Linguistics*, Columbus, Ohio, USA, pp. 780–88.

Appendix

A Detailed algorithm for extracting LGMG templates

We here describe in detail the algorithm for extracting LGMG templates, which was introduced in Section 5.2.3 as part of the GSL algorithm. The input is a sample corpus S of sentences from the Web containing anchor-sets that were extracted for the input template in the ASE phase. For each sentence $s \in S$, s is parsed by Minipar and each anchor was replaced by its corresponding variable name.

A.1 Step 1: extracting minimal spanning trees

For each graph $s \in S$, we find all the roots of the different minimal spanning trees for the anchor variables in s (templates that cannot be further reduced without omitting a variable). According to our empirical constraints every learned template is an extension of such a spanning tree in some graphs in S (see Section 5.2.4). We construct for each minimal spanning tree st_s, which was found in s, a string description of that tree, denoted $desc(st_s)$. The string description is constructed in such a way that isomorphic trees have the same description. We note that two roots of different templates may merge in the same root node in $CGR(S)$. However, these two roots differ in their index sets, since the support sets of their templates (the template occurrences in S) are different. For example, the templates '$X \xleftarrow{gen}$ acquisition $\xrightarrow{of} Y$' and '$Y \xleftarrow{gen}$ acquisition $\xrightarrow{by} X$', which both participate in an entailment relation with '$X \xleftarrow{subj}$ acquire $\xrightarrow{obj} Y$', share the same root 'acquisition' in $CGR(S)$. Yet, their support sets are different, as the templates occur in different sentences. Thus, the index set of the root 'acquisition' in each template is different. We construct a map H_S whose keys are the different tree descriptions $desc(st)$, where st denotes a set of isomorphic trees. The value of $H_S(desc(st))$ is the root node r of these isomorphic trees in $CGR(S)$ together with the set of indices of r from the various occurrences in S of all isomorphic trees associated with st. Every LGMG template originates from a set of isomorphic minimal spanning trees st and, thus, it is associated with one and only one string description. Hence, at the end of this step, the value of every entry in H_S, which consists of a root node r and a specific index set idx, corresponds to one and only one LGMG template. From here on, we address the set of isomorphic trees st as a single tree for ease of reading, since it is represented by a single root node with an index set.

A.2 Step 2: expanding the minimal spanning trees to LGMG templates

Every minimal spanning tree st, which is obtained from a $CGR(S)$ root node r and an associated index set idx (an H_S entry extracted in the previous step), is expanded to the maximal directed acyclic sub-graph rooted in r that has the same number of occurrences in S as st. To do this, we mark the root tree r and propagate markers on the vertices that are descendants of r. Before proceeding, we have to introduce the definition of *associated path*. Informally, each path in $CGR(S)$ is an aggregation of one or more associated paths in S.

Definition

We say that a path a in an individual tree $s \in S$ is associated with a path p from $CGR(S)$, if and only if:

1. For each vertex $v_a \in a$ there exists a vertex $v_p \in p$, such that $id(v_a) \in index(v_p)$. We then say that the vertex v_a *corresponds to* v_p.
2. Moreover, for each edge $e_a = (v_{a1}, v_{a2}, l) \in a$ there exists an edge with the same label from p, $e_p = (v_{p1}, v_{p2}, l)$, such that $(id(v_{a1}), id(v_{a2})) \in index(e_p)$. We say that the edge e_a *corresponds to* e_p.
3. In a, the vertices and edges appear in the same order as their corresponding vertices and edges in p.

Each directed path p in CGR(S) that begins in r is represented via a marker M, which holds information about:

- $v_e(M)$ – the end vertex of p.
- The path between r and $v_e(M)$, denoted by $path(M)$.
- The set $idx(M)$, which contains the indices of the end vertices of the paths associated with p.

We use a processing queue Q', which stores the markers corresponding to the paths that cannot be further expanded, and a queue Q, which contains markers for paths that may be further expanded. For each minimal spanning tree st obtained in the previous step with a root r in $CGR(S)$ and an index set of the root occurrences idx, the following algorithm is repeated:

1. $Q = \{\}, Q' = \{\}$.
2. Create a new marker M_r for which: $v_e(M_r) = r$, $idx(M_r) = idx$, $path(M) = \{r\}$.
3. Add M_r to Q.
4. While Q is not empty:
   (a) Flag=false.
   (b) Take the next marker in Q and assign it to M.
   (c) For each v', a child of $v_e(M)$ s.t. $(v_e(M), v') \in E_{cgr}$:
      i Create a new marker M' and initialize its fields in the following way:
      $v_e(M') = v'$
      $idx(M') = \{j | (i, j) \in index(v_e(M), v'), i \in idx(M)\}$
      $path(M') = append(path(M), v')$
      ii If $idx(M')$ contains $|idx|$ elements, add M' at the end of Q; Flag=true[13]
   (d) If Flag=false, add M at the end of Q'.

At the end of the algorithm, the markers in Q' represent the paths that form a template t rooted in r that occurs as many times as the minimal spanning tree st.

A.3 Algorithm efficiency

To simplify our reasoning, we will calculate the efficiency in the case where all parsed sentences in S are trees in which no slot variable appears more than once. This is the structure of vast majority of the resulting parsed sentences in S. We denote with $|MST|$ the number of the different minimal spanning trees in $CGR(S)$, with $|V_s|$ the number of vertices in a tree s, and with $|V_S|$ the number of vertices in the tree collection S. In the first step of the algorithm, we find the root of the spanning tree in every $s \in S$ and generate its textual description. This step can be performed in a time linear in the number of nodes in s, $O(|V_s|)$, and for all trees, in time $O(|V_S|)$. Using a hash table with ordered keys as an implementation of the map H_S, we can construct the different index sets of every root of minimal spanning tree in time $O(|V_S| \cdot log_2|V_S|)$. The extraction of the different pairs of root and index-set from H_S is in time $O(|MST|)$. In the second step, we extract an LGMG template

[13] This step ensures that we expand the paths while the size of their support set in S is equal to the size of the support set of the minimal spanning tree st, which is equal to $|idx|$.

for each root of a minimal spanning tree r by simultaneously tracing paths in $CGR(S)$ that start at r. The basic operation when tracing paths is the expansion of markers, in which the most time-consuming process is the calculation of the index-set of the expanded markers: $idx(M') = \{j|(i,j) \in index(v_e(M), v'), i \in idx(M)\}$. In this calculation, for each arc in S, denoted via pair of indices (i,j), we have to find if $i \in idx(M)$. If the index-sets are ordered, at most $log_2(|idx(M)|)$ comparisons are needed to do this.We note that $|idx(M)| < |V_S|$, the number of the vertices in S. In addition, since we consider tree structures, we pass at most once through each arc from S while expanding markers. Since the number of arcs is less than $|V_S|$, the number of comparisons for constructing each template is $O(|V_S| \cdot log_2|V_S|)$. Using $|MST|$ as the number of the minimal spanning trees, we obtain a time complexity of $O(|MST| \cdot |V_S| \cdot log_2|V_S|)$. Taking into account the time complexity estimation of the two algorithm steps, we obtain a bound of $O(|V_S| + |MST| + |MST| \cdot |V_S| \cdot log|V_S|) = O(|MST| \cdot |V_S| \cdot log_2|V_S|)$. The number of the minimal sub-trees that span over variables is bounded by the size of the language lexicon, by the number of types of syntactic relations and the maximal depth of the syntactic constructions. In any natural language, these parameters are bounded by constants; therefore, the number of possible spanning trees can be considered bounded by a constant. Considering this, we can ignore $|MST|$ in the complexity formula and obtain as a final complexity the estimate $O(|V_S| \cdot log_2|V_S|)$.