Learning Entailment Relations by Global Graph Structure Optimization

Jonathan Berant*
Tel Aviv University

Ido Dagan**
Bar-Ilan University

Jacob Goldberger†
Bar-Ilan University

Identifying entailment relations between predicates is an important part of applied semantic inference. In this article we propose a global inference algorithm that learns such entailment rules. First, we define a graph structure over predicates that represents entailment relations as directed edges. Then, we use a global transitivity constraint on the graph to learn the optimal set of edges, formulating the optimization problem as an Integer Linear Program. The algorithm is applied in a setting where, given a target concept, the algorithm learns on the fly all entailment rules between predicates that co-occur with this concept. Results show that our global algorithm improves performance over baseline algorithms by more than 10%.

1. Introduction

The **Textual Entailment (TE)** paradigm is a generic framework for applied semantic inference. The objective of TE is to recognize whether a target textual meaning can be inferred from another given text. For example, a question answering system has to recognize that *alcohol affects blood pressure* is inferred from the text *alcohol reduces blood pressure* to answer the question *What affects blood pressure?* In the TE framework, entailment is defined as a directional relationship between pairs of text expressions, denoted by T, the entailing text, and H, the entailed hypothesis. The text T is said to entail the hypothesis H if, typically, a human reading T would infer that H is most likely true (Dagan et al. 2009).

TE systems require extensive knowledge of entailment patterns, often captured as **entailment rules**—rules that specify a directional inference relation between two text fragments (when the rule is bidirectional this is known as **paraphrasing**). A common type of text fragment is a **proposition**, which is a simple natural language expression that contains a **predicate** and **arguments** (such as *alcohol affects blood pressure*), where the predicate denotes some semantic relation between the **concepts** that are expressed

* Tel-Aviv University, P.O. Box 39040, Tel-Aviv, 69978, Israel. E-mail: jonatha6@post.tau.ac.il.
** Bar-Ilan University, Ramat-Gan, 52900, Israel. E-mail: dagan@cs.biu.ac.il.
† Bar-Ilan University, Ramat-Gan, 52900, Israel. E-mail: goldbej@eng.biu.ac.il.

Submission received: 28 September 2010; revised submission received: 5 May 2011; accepted for publication: 5 July 2011.

by the arguments. One important type of entailment rule specifies entailment between **propositional templates**, that is, propositions where the arguments are possibly replaced by variables. A rule corresponding to the aforementioned example may be *X reduce blood pressure* $\rightarrow$ *X affect blood pressure*. Because facts and knowledge are mostly expressed by propositions, such entailment rules are central to the TE task. This has led to active research on broad-scale acquisition of entailment rules for predicates (Lin and Pantel 2001; Sekine 2005; Szpektor and Dagan 2008; Yates and Etzioni 2009; Schoenmackers et al. 2010).

Previous work has focused on learning each entailment rule in isolation. It is clear, however, that there are interactions between rules. A prominent phenomenon is that entailment is inherently a transitive relation, and thus the rules $X \rightarrow Y$ and $Y \rightarrow Z$ imply the rule $X \rightarrow Z$.[1] In this article we take advantage of these global interactions to improve entailment rule learning.

After reviewing relevant background (Section 2), we describe a structure termed an **entailment graph** that models entailment relations between propositional templates (Section 3). Next, we motivate and discuss a specific type of entailment graph, termed a **focused entailment graph**, where a **target concept** instantiates one of the arguments of all propositional templates. For example, a focused entailment graph about the target concept *nausea* might specify the entailment relations between propositional templates like *X induce nausea*, *X prevent nausea*, and *nausea is a symptom of X*.

In the core section of the article, we present an algorithm that uses a global approach to learn the entailment relations, which comprise the edges of focused entailment graphs (Section 4). We define a global objective function and look for the graph that maximizes that function given scores provided by a local entailment classifier and a global transitivity constraint. The optimization problem is formulated as an **Integer Linear Program (ILP)** and is solved with an ILP solver, which leads to an optimal solution with respect to the global function. In Section 5 we demonstrate that this algorithm outperforms by 12–13% methods that utilize only local information as well as methods that employ a greedy optimization algorithm (Snow, Jurafsky, and Ng 2006) rather than an ILP solver.

The article also includes a comprehensive investigation of the algorithm and its components. First, we perform manual comparison between our algorithm and the baselines and analyze the reasons for the improvement in performance (Sections 5.3.1 and 5.3.2). Then, we analyze the errors made by the algorithm against manually prepared gold-standard graphs and compare them to the baselines (Section 5.4). Last, we perform a series of experiments in which we investigate the local entailment classifier and specifically experiment with various sets of features (Section 6). We conclude and suggest future research directions in Section 7.

This article is based on previous work (Berant, Dagan, and Goldberger 2010), while substantially expanding upon it. From a theoretical point of view, we reformulate the two ILPs previously introduced by incorporating a prior. We show a theoretical relation between the two ILPs and prove that the optimization problem tackled is NP-hard. From an empirical point of view, we conduct many new experiments that examine both the local entailment classifier as well as the global algorithm. Last, a rigorous analysis of the algorithm is performed and an extensive survey of previous work is provided.

1 Assuming that Y has the same sense in both $X \rightarrow Y$ and $Y \rightarrow Z$, as we discuss later in Section 3.

2. Background

In this section we survey methods proposed in past literature for learning entailment rules between predicates. First, we discuss local methods that assess entailment given a pair of predicates, and then global methods that perform inference over a larger set of predicates.

2.1 Local Learning

Three types of information have primarily been utilized in the past to learn entailment rules between predicates: lexicographic methods, distributional similarity methods, and pattern-based methods.

Lexicographic methods use manually prepared knowledge bases that contain information about semantic relations between lexical items. WordNet (Fellbaum 1998b), by far the most widely used resource, specifies relations such as hyponymy, synonymy, **derivation**, and **entailment** that can be used for semantic inference (Budanitsky and Hirst 2006). For example, if WordNet specifies that *reduce* is a hyponym of *affect*, then one can infer that *X reduces Y* $\rightarrow$ *X affects Y*. WordNet has also been exploited to automatically generate a training set for a hyponym classifier (Snow, Jurafsky, and Ng 2004), and we make a similar use of WordNet in Section 4.1.

A drawback of WordNet is that it specifies semantic relations for words and terms but not for more complex expressions. For example, WordNet does not cover a complex predicate such as *X causes a reduction in Y*. Another drawback of WordNet is that it only supplies semantic relations between lexical items, but does not provide any information on how to map arguments of predicates. For example, WordNet specifies that there is an entailment relation between the predicates *pay* and *buy*, but does not describe the way in which arguments are mapped: if *X pays Y for Z* then *X buys Z from Y*. Thus, using WordNet directly to derive entailment rules between predicates is possible only for semantic relations such as hyponymy and synonymy, where arguments typically preserve their syntactic positions on both sides of the rule.

Some knowledge bases try to overcome this difficulty: Nomlex (Macleod et al. 1998) is a dictionary that provides the mapping of arguments between verbs and their nominalizations and has been utilized to derive predicative entailment rules (Meyers et al. 2004; Szpektor and Dagan 2009). FrameNet (Baker, Fillmore, and Lowe 1998) is a lexicographic resource that is arranged around "frames": Each frame corresponds to an event and includes information on the predicates and arguments relevant for that specific event supplemented with annotated examples that specify argument positions. Consequently, FrameNet was also used to derive entailment rules between predicates (Coyne and Rambow 2009; Ben Aharon, Szpektor, and Dagan 2010). Additional manually constructed resources for predicates include PropBank (Kingsbury, Palmer, and Marcus 2002) and VerbNet (Kipper, Dang, and Palmer 2000).

Distributional similarity methods are used to learn broad-scale resources, because lexicographic resources tend to have limited coverage. Distributional similarity algorithms employ "the distributional hypothesis" (Harris 1954) and predict a semantic relation between two predicates by comparing the arguments with which they occur. Quite a few methods have been suggested (Lin and Pantel 2001; Szpektor et al. 2004; Bhagat, Pantel, and Hovy 2007; Szpektor and Dagan 2008; Yates and Etzioni 2009; Schoenmackers et al. 2010), which differ in terms of the specifics of the ways in which predicates are represented, the features that are extracted, and the function used to compute feature vector similarity. Next, we elaborate on some of the prominent methods.

Lin and Pantel (2001) proposed an algorithm that is based on a mutual information criterion. A predicate is represented by a **binary template**, which is a dependency path between two arguments of a predicate where the arguments are replaced by variables. Note that in a dependency tree, a path between two arguments must pass through their common predicate. Also note that if a predicate has more than two arguments, then it is represented by more than one binary template, where each template corresponds to a different aspect of the predicate. For example, the proposition *I bought a gift for her* contains a predicate and three arguments, and therefore is represented by the following three binary templates: $X \xleftarrow{subj} buys \xrightarrow{obj} Y$, $X \xleftarrow{obj} buys \xrightarrow{prep} for \xrightarrow{pcomp-n} Y$ and $X \xleftarrow{subj} buys \xrightarrow{prep} for \xrightarrow{pcomp-n} Y$.

For each binary template Lin and Pantel compute two sets of features F_x and F_y, which are the words that instantiate the arguments X and Y, respectively, in a large corpus. Given a template t and its feature set for the X variable F_x^t, every $f_x \in F_x^t$ is weighted by the pointwise mutual information between the template and the feature: $w_x^t(f_x) = \log \frac{Pr(f_x|t)}{Pr(f_x)}$, where the probabilities are computed using maximum likelihood over the corpus. Given two templates u and v, the *Lin* measure (Lin 1998a) is computed for the variable X in the following manner:

$$Lin_x(u,v) = \frac{\sum_{f \in F_x^u \cap F_x^v} [w_x^u(f) + w_x^v(f)]}{\sum_{f \in F_x^u} w_x^u(f) + \sum_{f \in F_x^v} w_x^v(f)} \quad (1)$$

The measure is computed analogously for the variable Y and the final distributional similarity score, termed *DIRT*, is the geometric average of the scores for the two variables:

$$DIRT(u,v) = \sqrt{Lin_x(u,v) \cdot Lin_y(u,v)} \quad (2)$$

If $DIRT(u,v)$ is high, this means that the templates u and v share many "informative" arguments and so it is possible that $u \rightarrow v$. Note, however, that the *DIRT* similarity measure computes a symmetric score, which is appropriate for modeling synonymy but not entailment, an inherently directional relation.

To remedy that, Szpektor and Dagan (2008) suggested a directional distributional similarity measure. In their work, Szpektor and Dagan chose to represent predicates with **unary templates**, which are identical to binary templates, only they contain a predicate and a single argument, such as: $X \xleftarrow{subj} buys$. Szpektor and Dagan explain that unary templates are more expressive than binary templates, and that some predicates can only be encoded using unary templates. They propose that if for two unary templates $u \rightarrow v$, then relatively many of the features of u should be covered by the features of v. This is captured by the asymmetric *Cover* measure suggested by Weeds and Weir (2003) (we omit the subscript x from F_x^u and F_x^v because in their setting there is only one argument):

$$Cover(u,v) = \frac{\sum_{f \in F^u \cap F^v} w^u(f)}{\sum_{f \in F^u} w^u(f)} \quad (3)$$

The final directional score, termed *BInc* (Balanced Inclusion), is the geometric average of the *Lin* measure and the *Cover* measure:

$$BInc(u,v) = \sqrt{Lin(u,v) \cdot Cover(u,v)} \quad (4)$$

Both Lin and Pantel as well as Szpektor and Dagan compute a similarity score for each argument separately, effectively decoupling the arguments from one another. It is clear, however, that although this alleviates sparsity problems, it disregards an important piece of information, namely, the co-occurrence of arguments. For example, if one looks at the following propositions: *coffee increases blood pressure*, *coffee decreases fatigue*, *wine decreases blood pressure*, *wine increases fatigue*, one can notice that the predicates occur with similar arguments and might mistakenly infer that *decrease* $\rightarrow$ *increase*. However, looking at pairs of arguments reveals that the predicates do not share a single pair of arguments.

Yates and Etzioni (2009) address this issue and propose a generative model that estimates the probability that two predicates are synonymous (synonymy is simply bidirectional entailment) by comparing pairs of arguments. They represent predicates and arguments as strings and compute for every predicate a feature vector that counts that number of times it occurs with any ordered pair of words as arguments. Their main modeling decision is to assume that two predicates are synonymous if the number of pairs of arguments they share is maximal. An earlier work by Szpektor et al. (2004) also tried to learn entailment rules between predicates by using pairs of arguments as features. They utilized an algorithm that learns new rules by searching for distributional similarity information on the Web for candidate predicates.

Pattern-based methods. Although distributional similarity measures excel at identifying the existence of semantic similarity between predicates, they are often unable to discern the exact type of semantic similarity and specifically determine whether it is entailment. Pattern-based methods are used to automatically extract pairs of predicates for a specific semantic relation. Pattern-based methods identify a semantic relation between two predicates by observing that they co-occur in specific patterns in sentences. For example, from the single proposition *He scared and even startled me* one might infer that *startle* is semantically stronger than *scare* and thus *startle* $\rightarrow$ *scare*. Chklovski and Pantel (2004) manually constructed a few dozen patterns and learned semantic relations between predicates by looking for these patterns on the Web. For example, the pattern *X and even Y* implies that *Y* is stronger than *X*, and the pattern *to X and then Y* indicates that *Y* follows *X*. The main disadvantage of pattern-based methods is that they are based on the co-occurrence of two predicates in a single sentence in a specific pattern. These events are quite rare and require working on a very large corpus, or preferably, the Web.

Pattern-based methods were mainly utilized so far to extract semantic relations between nouns, and there has been some work on automatically learning patterns for nouns (Snow, Jurafsky, and Ng 2004). Although these methods can be expanded for predicates, we are unaware of any attempt to automatically learn patterns that describe semantic relations between predicates (as opposed to the manually constructed patterns suggested by Chklovski and Pantel [2004]).

2.2 Global Learning

It is natural to describe entailment relations between predicates (or language expressions in general) by a graph. Nodes represent predicates, and edges represent entailment between nodes. Nevertheless, using a graph for global learning of all entailment relations within a set of predicates, rather then between pairs of predicates, has attracted little attention. Recently, Szpektor and Dagan (2009) presented the resource Argument-mapped WordNet, providing entailment relations for predicates in WordNet. This resource was built on top of WordNet and augments it with mapping of arguments for predicates using NomLex (Macleod et al. 1998) and a corpus-based resource (Szpektor

and Dagan 2008). Their resource makes simple use of WordNet's global graph structure: New rules are suggested by transitively chaining graph edges, and then verified using distributional similarity measures. Effectively, this is equivalent to using the intersection of the set of rules derived by this transitive chaining and the set of rules in a distributional similarity knowledge base.

The most similar work to ours is Snow, Jurafsky, and Ng's (2006) algorithm for taxonomy induction, although it involves learning the hyponymy relation between nouns, which is a special case of entailment, rather than learning entailment between predicates. We provide here a brief review of a simplified form of this algorithm.

Snow, Jurafsky, and Ng define a taxonomy T to be a set of pairs of words, expressing the hyponymy relation between them. The notation $H_{uv} \in T$ means that the noun u is a hyponym of the noun v in T. They define D to be the set of observed data over all pairs of words, and define $D_{uv} \in D$ to be the observed evidence we have in the data for the event $H_{uv} \in T$. Snow, Jurafsky, and Ng assume a model exists for inferring $P(H_{uv} \in T|D_{uv})$: the posterior probability of the event $H_{uv} \in T$, given the data. Their goal is to find the taxonomy that maximizes the likelihood of the data, that is, to find

$$\hat{T} = \underset{T}{\operatorname{argmax}} P(D|T) \tag{5}$$

Using some independence assumptions and Bayes rule, the likelihood $P(D|T)$ is expressed:

$$P(D|T) = \prod_{H_{uv} \in T} \frac{P(H_{uv} \in T|D_{uv})P(D_{uv})}{P(H_{uv} \in T)} \cdot \prod_{H_{uv} \notin T} \frac{P(H_{uv} \notin T|D_u v)P(D_{uv})}{P(H_{uv} \notin T)} \tag{6}$$

Crucially, they demand that the taxonomy learned respects the constraint that hyponymy is a transitive relation. To ensure that, they propose the following greedy algorithm: At each step they go over all pairs of words (u, v) that are not in the taxonomy, and try to add the single hyponymy relation H_{uv}. Then, they calculate the set of relations S_{uv} that H_{uv} will add to the taxonomy due to the transitivity constraint (all of the relations H_{uw}, where w is a hypernym of v in the taxonomy). Last, they choose to add that set of relations S_{uv} that maximizes $P(D|T)$ out of all the possible candidates. This iterative process stops when $P(D|T)$ starts dropping. Their implementation of the algorithm uses a hyponym classifier presented in an earlier work (Snow, Jurafsky, and Ng 2004) as a model for $P(H_{uv} \in T|D_{uv})$ and a single *sparsity* parameter $k = \frac{P(H_{uv} \notin T)}{P(H_{uv} \in T)}$. In this article we tackle a similar problem of learning a transitive relation, but we use linear programming (Vanderbei 2008) to solve the optimization problem.

2.3 Linear Programming

A **Linear Program (LP)** is an optimization problem where a linear objective function is minimized (or maximized) under linear constraints.

$$\begin{aligned} &\min_{x \in \mathbb{R}^d} c^\top x \\ &\text{such that } Ax \leq b \end{aligned} \tag{7}$$

where $c \in \mathbb{R}^d$ is a coefficient vector, and $A \in \mathbb{R}^n \times \mathbb{R}^d$ and $b \in \mathbb{R}^n$ specify the constraints. In short, we wish to find the optimal assignment for the d variables in the vector x, such

that all n linear constraints specified by the matrix A and the vector b are satisfied by this assignment. If the variables are forced to be integers, the problem is termed an Integer Linear Program (ILP). ILP has attracted considerable attention recently in several fields of NLP, such as semantic role labeling, summarization, and parsing (Althaus, Karamanis, and Koller 2004; Roth and Yih 2004; Riedel and Clarke 2006; Clarke and Lapata 2008; Finkel and Manning 2008; Martins, Smith, and Xing 2009). In this article we formulate the entailment graph learning problem as an ILP, which leads to an optimal solution with respect to the objective function (vs. a greedy optimization algorithm suggested by Snow, Jurafsky, and Ng [2006]). Recently, Do and Roth (2010) used ILP in a related task of learning taxonomic relations between nouns, utilizing constraints between sibling nodes and ancestor–child nodes in small graphs of three nodes.

3. Entailment Graph

In this section we define a structure termed the **entailment graph** that describes the entailment relations between propositional templates (Section 3.1), and a specific type of entailment graph, termed the **focused entailment graph**, that concentrates on entailment relations that are relevant for some pre-defined target concept (Section 3.2).

3.1 Entailment Graph: Definition and Properties

The nodes of an entailment graph are **propositional templates**. A propositional template is a binary template[2] where at least one of the two arguments is a variable whereas the second may be instantiated. In addition, the sense of the predicate is specified (according to some sense inventory, such as WordNet) and so each sense of a polysemous predicate corresponds to a separate template (and a separate graph node). For example, $X \xleftarrow{subj} treats\#1 \xrightarrow{obj} Y$ and $X \xleftarrow{subj} treats\#2 \xrightarrow{obj} nausea$ are propositional templates for the first and second sense of the predicate *treat*, respectively. An edge (u, v) represents the fact that template u entails template v. Note that the entailment relation transcends hyponymy/troponomy. For example, the template *X is diagnosed with asthma* entails the template *X suffers from asthma*, although one is not a hyponym of the other. An example of an entailment graph is given in Figure 1.

Because entailment is a transitive relation, an entailment graph is **transitive**, that is, if the edges (u, v) and (v, w) are in the graph, so is the edge (u, w). Note that the property of transitivity does not hold when the senses of the predicates are not specified. For example, *X buys Y* $\rightarrow$ *X acquires Y* and *X acquires Y* $\rightarrow$ *X learns Y*, but *X buys Y* $\nrightarrow$ *X learns Y*. This violation occurs because the predicate *acquire* has two distinct senses in the two templates, but this distinction is lost when senses are not specified.

Transitivity implies that in each strongly connected component[3] of the graph all nodes entail each other. For example, in Figure 1 the nodes *X-related-to-nausea* and *X-associated-with-nausea* form a strongly connected component. Moreover, if we merge every strongly connected component to a single node, the graph becomes a Directed Acyclic Graph (DAG), and a hierarchy of predicates can be obtained.

2 We restrict our discussion to templates with two arguments, but generalization is straightforward.
3 A strongly connected component is a subset of nodes in the graph where there is a path from any node to any other node.

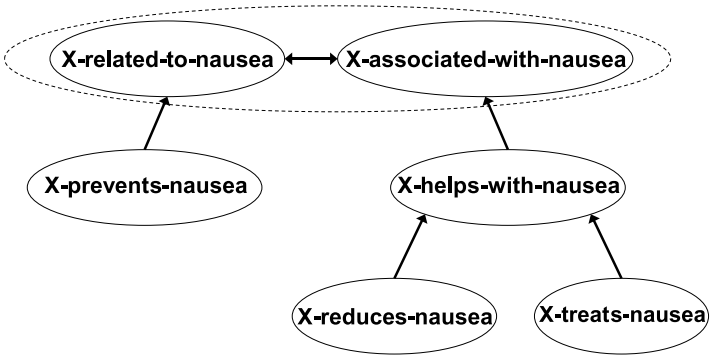

Figure 1
A focused entailment graph. For clarity, edges that can be inferred by transitivity are omitted. The single strongly connected component is surrounded by a dashed line.

3.2 Focused Entailment Graphs

In this article we concentrate on learning a type of entailment graph, termed the focused entailment graph. Given a target concept, such as *nausea*, a focused entailment graph describes the entailment relations between propositional templates for which the target concept is one of the arguments (see Figure 1). Learning such entailment rules in real time for a target concept is useful in scenarios such as information retrieval and question answering, where a user specifies a query about the target concept. The need for such rules has been also motivated by Clark et al. (2007), who investigated what types of knowledge are needed to identify entailment in the context of the RTE challenge, and found that often rules that are specific to a certain concept are required. Another example for a semantic inference algorithm that is utilized in real time is provided by Do and Roth (2010), who recently described a system that, given two terms, determines the taxonomic relation between them on the fly. Last, we have recently suggested an application that uses focused entailment graphs to present information about a target concept according to a hierarchy of entailment (Berant, Dagan, and Goldberger 2010).

The benefit of learning focused entailment graphs is three-fold. First, the target concept that instantiates the propositional template usually disambiguates the predicate and hence the problem of predicate ambiguity is greatly reduced. Thus, we do not employ any form of disambiguation in this article, but assume that every node in a focused entailment graph has a single sense (we further discuss this assumption when describing the experimental setting in Section 5.1), which allows us to utilize transitivity constraints.

An additional (albeit rare) reason that might also cause violations of transitivity constraints is the notion of **probabilistic entailment**. Whereas troponomy rules (Fellbaum 1998a) such as *X walks* $\rightarrow$ *X moves* can be perceived as being almost always correct, rules such as *X coughs* $\rightarrow$ *X is sick* might only be true with some probability. Consequently, chaining a few probabilistic rules such as $A \rightarrow B$, $B \rightarrow C$, and $C \rightarrow D$ might not guarantee the correctness of $A \rightarrow D$. Because in focused entailment graphs the number of nodes and diameter[4] are quite small (for example, in the data set we

4 The distance between two nodes in a graph is the number of edges in a shortest path connecting them. The diameter of a graph is the maximal distance between any two nodes in the graph.

present in Section 5 the maximal number of nodes is 26, the average number of nodes is 22.04, the maximal diameter is 5, and the average diameter is 2.44), we do not find this to be a problem in our experiments in practice.

Last, the optimization problem that we formulate is NP-hard (as we show in Section 4.2). Because the number of nodes in focused entailment graphs is rather small, a standard ILP solver is able to quickly reach the optimal solution.

To conclude, the algorithm we suggest next is applied in our experiments on focused entailment graphs. However, we believe that it is suitable for any entailment graph whose properties are similar to those of focused entailment graphs. For brevity, from now on the term *entailment graph* will stand for *focused entailment graph*.

4. Learning Entailment Graph Edges

In this section we present an algorithm that, given the set of propositional templates constituting the nodes of an entailment graph, learns its edges (i.e., the entailment relations between all pairs of nodes). The algorithm comprises two steps (described in Sections 4.1 and 4.2): In the first step we use a large corpus and a lexicographic resource (WordNet) to train a generic **entailment classifier** that given any pair of propositional templates estimates the likelihood that one template entails the other. This generic step is performed only once, and is independent of the specific nodes of the target entailment graph whose edges we want to learn. In the second step we learn on the fly the edges of a specific target graph: Given the graph nodes, we use a global optimization approach that determines the set of edges that maximizes the probability (or score) of the entire graph. The global graph decision is determined by the given edge probabilities (or scores) supplied by the entailment classifier and by the graph constraints (transitivity and others).

4.1 Training an Entailment Classifier

We describe a procedure for learning a generic entailment classifier, which can be used to estimate the entailment likelihood for any given pair of templates. The classifier is constructed based on a corpus and a lexicographic resource (WordNet) using the following four steps:

(1) Extract a large set of propositional templates from the corpus.

(2) Use WordNet to automatically generate a training set of pairs of templates—both positive and negative examples.

(3) Represent each training set example with a feature vector of various distributional similarity scores.

(4) Train a classifier over the training set.

(1) Template extraction. We parse the corpus with the Minipar dependency parser (Lin 1998b) and use the Minipar representation to extract all binary templates from every parse tree, employing the procedure described by Lin and Pantel (2001), which considers all dependency paths between every pair of nouns in the parse tree. We also apply over the extracted paths the syntactic normalization procedure described by Szpektor and Dagan (2007), which includes transforming passive forms into active forms and removal of conjunctions, appositions, and abbreviations. In addition, we use

Table 1
Positive and negative examples for entailment in the training set. The direction of entailment is from the left template to the right template.

Positive examples	Negative examples
$(X \xleftarrow{subj} desires \xrightarrow{obj} Y, X \xleftarrow{subj} wants \xrightarrow{obj} Y)$	$(X \xleftarrow{subj} pushes \xrightarrow{obj} Y, X \xleftarrow{subj} blows \xrightarrow{obj} Y)$
$(X \xleftarrow{subj} causes \xleftarrow{vrel} Y, X \xleftarrow{subj} creates \xleftarrow{vrel} Y)$	$(X \xleftarrow{subj} issues \xleftarrow{vrel} Y, X \xleftarrow{subj} signs \xleftarrow{vrel} Y)$

a simple heuristic to filter out templates that probably do not include a predicate: We omit "uni-directional" templates where the root of template has a single child, such as $therapy \xrightarrow{prep} in \xrightarrow{p-comp} patient \xrightarrow{nn} cancer$, unless one of the edges is labeled with a passive relation, such as in the template $nausea \xleftarrow{vrel} characterized \xleftarrow{subj} poisoning$, which contains the Minipar passive label *vrel*.[5] Last, the arguments are replaced by variables, resulting in propositional templates such as $X \xleftarrow{subj} affect \xrightarrow{obj} Y$. The lexical items that remain in the template after replacing the arguments by variables are termed **predicate words**.

(2) Training set generation. WordNet is used to automatically generate a training set of positive (entailing) and negative (non-entailing) template pairs. Let T be the set of propositional templates extracted from the corpus. For each $t_i \in T$ with two variables and a single predicate word w, we extract from WordNet the set H of direct hypernyms (distance of one in WordNet) and synonyms of w. For every $h \in H$, we generate a new template t_j from t_i by replacing w with h. If $t_j \in T$, we consider (t_i, t_j) to be a positive example. Negative examples are generated analogously, only considering direct co-hyponyms of w, which are direct hyponyms of direct hypernyms of w that are not synonymous to w. It has been shown in past work that in most cases co-hyponym terms do not entail one another (Mirkin, Dagan, and Gefet 2006). A few examples for positive and negative training examples are given in Table 1.

This generation method is similar to the "distant supervision" method proposed by Snow, Jurafsky, and Ng (2004) for training a noun hypernym classifier. It differs in some important aspects, however: First, Snow, Jurafsky, and Ng consider a positive example to be any Wordnet hypernym, irrespective of the distance, whereas we look only at direct hypernyms. This is because predicates are mainly verbs and precision drops quickly when looking at verb hypernyms in WordNet at a longer distance. Second, Snow, Jurafsky, and Ng generate negative examples by looking at any two nouns where one is not the hypernym of the other. In the spirit of "contrastive estimation" (Smith and Eisner 2005), we prefer to generate negative examples that are "hard," that is, negative examples that, although not entailing, are still semantically similar to positive examples and thus focus the classifier's attention on determining the boundary of the entailment class. Last, we use a balanced number of positive and negative examples, because classifiers tend to perform poorly on the minority class when trained on imbalanced data (Van Hulse, Khoshgoftaar, and Napolitano 2007; Nikulin 2008).

(3) Distributional similarity representation. We aim to train a classifier that for an input template pair (t_1, t_2) determines whether t_1 entails t_2. Our approach is to represent a template pair by a feature vector where each coordinate is a different distributional similarity score for the pair of templates. The different distributional similarity scores

5 This passive construction is not handled by the normalization scheme employed by Szpektor and Dagan (2007).

are obtained by utilizing various distributional similarity algorithms that differ in one or more of their characteristics. In this way we hope to combine the various methods proposed in the past for measuring distributional similarity. The distributional similarity algorithms we employ vary in one or more of the following dimensions: the way the predicate is represented, the way the features are represented, and the function used to measure similarity between the feature representations of the two templates.

Predicate representation. As mentioned, we represent predicates over dependency tree structures. However, some distributional similarity algorithms measure similarity between binary templates directly (Lin and Pantel 2001; Szpektor et al. 2004; Bhagat, Pantel, and Hovy 2007; Yates and Etzioni 2009), whereas others decompose binary templates into two unary templates, estimate similarity between two pairs of unary templates, and combine the two scores into a single score (Szpektor and Dagan 2008).

Feature representation. The features of a template are some function of the terms that instantiated the argument variables in a corpus. Two representations that are used in our experiments are derived from an ontology that maps natural language phrases to semantic identifiers (see Section 5). Another variant occurs when using binary templates: a template may be represented by a pair of feature vectors, one for each variable as in the DIRT algorithm (Lin and Pantel 2001), or by a single vector, where features represent pairs of instantiations (Szpektor et al. 2004; Yates and Etzioni 2009). The former variant reduces sparsity problems, whereas Yates and Etzioni showed the latter is more informative and performs favorably on their data.

Similarity function. We consider two similarity functions: The symmetric *Lin* (Lin and Pantel 2001) similarity measure, and the directional *BInc* (Szpektor and Dagan 2008) similarity measure, reviewed in Section 2. Thus, information about the direction of entailment is provided by the *BInc* measure.

We compute for any pair of templates (t_1, t_2) 12 distributional similarity scores using all possible combinations of the aforementioned dimensions. These scores are then used as 12 features representing the pair (t_1, t_2). (A full description of the features is given in Section 5.) This is reminiscent of Connor and Roth (2007), who used the output of unsupervised classifiers as features for a supervised classifier in a verb disambiguation task.

(4) Training a classifier Two types of classifiers may be trained in our scheme over the training set: margin classifiers (such as SVM) and probabilistic classifiers. Given a pair of templates (u, v) and their feature vector F_{uv}, we denote by an indicator variable I_{uv} the event that u entails v. A margin classifier estimates a score S_{uv} for the event $I_{uv} = 1$, which indicates the positive or negative distance of the feature vector F_{uv} from the separating hyperplane. A probabilistic classifier provides the posterior probability $P_{uv} = P(I_{uv} = 1|F_{uv})$.

4.2 Global Learning of Edges

In this step we get a set of propositional templates as input, and we would like to learn all of the entailment relations between these propositional templates. For every pair of templates we can compute the distributional similarity features and get a score from the trained entailment classifier. Once all the scores are calculated we try to find the optimal graph—that is, the best set of edges over the propositional templates. Thus, in this scenario the input is the nodes of the graph and the output are the edges.

To learn edges we consider global constraints, which allow only certain graph topologies. Because we seek a global solution under transitivity and other constraints, ILP is a natural choice, enabling the use of state-of-the-art ILP optimization packages. Given a set of nodes V and a weighting function $f : V \times V \rightarrow \mathbb{R}$ (derived from the

entailment classifier in our case), we want to learn the directed graph $G = (V, E)$, where $E = \{(u, v) | \ I_{uv} = 1\}$, by solving the following ILP over the variables I_{uv}:

$$\hat{G} = \underset{G}{\text{argmax}} \sum_{u \neq v} f(u, v) \cdot I_{uv} \tag{8}$$

$$\text{s.t. } \forall_{u,v,w \in V} \ I_{uv} + I_{vw} - I_{uw} \leq 1 \tag{9}$$

$$\forall_{u,v \in A_{yes}} \ I_{uv} = 1 \tag{10}$$

$$\forall_{u,v \in A_{no}} \ I_{uv} = 0 \tag{11}$$

$$\forall_{u \neq v} \ I_{uv} \in \{0, 1\} \tag{12}$$

The objective function in Equation (8) is simply a sum over the weights of the graph edges. The global constraint is given in Equation (9) and states that the graph must respect transitivity. This constraint is equivalent to the one suggested by Finkel and Manning (2008) in a coreference resolution task, except that the edges of our graph are directed. The constraints in Equations (10) and (11) state that for a few node pairs, defined by the sets A_{yes} and A_{no}, respectively, we have prior knowledge that one node does or does not entail the other node. Note that if $(u, v) \in A_{no}$, then due to transitivity there must be no path in the graph from u to v, which rules out additional edge combinations. We elaborate on how the sets A_{yes} and A_{no} are computed in our experiments in Section 5. Altogether, this Integer Linear Program contains $O(|V|^2)$ variables and $O(|V|^3)$ constraints, and can be solved using state-of-the-art optimization packages.

A theoretical aspect of this optimization problem is that it is NP-hard. We can phrase it as a decision problem in the following manner: Given V, f, and a threshold k, we wish to know if there is a set of edges E that respects transitivity and $\sum_{(u,v) \in E} f(u, v) \geq k$. Yannakakis (1978) has shown that the simpler problem of finding in a graph $G' = (V', E')$ a subset of edges $A \subseteq E'$ that respects transitivity and $|A| \geq k$ is NP-hard. Thus, we can conclude that our optimization problem is also NP-hard by the trivial polynomial reduction defining the function f that assigns the score 0 for node pairs $(u, v) \notin E'$ and the score 1 for node pairs $(u, v) \in E'$. Because the decision problem is NP-hard, it is clear that the corresponding maximization problem is also NP-hard. Thus, obtaining a solution using ILP is quite reasonable and in our experiments also proves to be efficient (Section 5).

Next, we describe two ways of obtaining the weighting function f, depending on the type of entailment classifier we prefer to train.

4.2.1 Score-Based Weighting Function. In this case, we assume that we choose to train a margin entailment classifier estimating the score S_{uv} (a positive score if the classifier predicts entailment, and a negative score otherwise) and define $f_{score}(u, v) = S_{uv} - \lambda$. This gives rise to the following objective function:

$$\hat{G}_{score} = \underset{G}{\text{argmax}} \sum_{u \neq v} (S_{uv} - \lambda) \cdot I_{uv} = \underset{G}{\text{argmax}} \left(\sum_{u \neq v} S_{uv} \cdot I_{uv} \right) - \lambda \cdot |E| \tag{13}$$

The term $\lambda \cdot |E|$ is a regularization term reflecting the fact that edges are sparse. Intuitively, this means that we would like to insert into the graph only edges with a score

$S_{uv} > \lambda$, or in other words to "push" the separating hyperplane towards the positive half space by λ. Note that the constant λ is a parameter that needs to be estimated and we discuss ways of estimating it in Section 5.2.

4.2.2 Probabilistic Weighting Function. In this case, we assume that we choose to train a probabilistic entailment classifier. Recall that I_{uv} is an indicator variable denoting whether u entails v, that F_{uv} is the feature vector for the pair of templates u and v, and define F to be the set of feature vectors for all pairs of templates in the graph. The classifier estimates the posterior probability of an edge given its features: $P_{uv} = P(I_{uv} = 1|F_{uv})$, and we would like to look for the graph G that maximizes the posterior probability $P(G|F)$. In Appendix A we specify some simplifying independence assumptions under which this graph maximizes the following linear objective function:

$$\hat{G}_{prob} = \underset{G}{\text{argmax}} \sum_{u \neq v} (\log \frac{P_{uv}}{1 - P_{uv}} + \log \eta) \cdot I_{uv} = \underset{G}{\text{argmax}} \sum_{u \neq v} \log \frac{P_{uv}}{1 - P_{uv}} \cdot I_{uv} + \log \eta \cdot |E| \tag{14}$$

where $\eta = \frac{P(I_{uv}=1)}{P(I_{uv}=0)}$ is the prior odds ratio for an edge in the graph, which needs to be estimated in some manner. Thus, the weighting function is defined by $f_{prob}(u, v) = \log \frac{P_{uv}}{1-P_{uv}} + \log \eta$.

Both the score-based and the probabilistic objective functions obtained are quite similar: Both contain a weighted sum over the edges and a regularization component reflecting the sparsity of the graph. Next, we show that we can provide a probabilistic interpretation for our score-based function (under certain conditions), which will allow us to use a margin classifier and interpret its output probabilistically.

4.2.3 Probabilistic Interpretation of Score-Based Weighting Function. We would like to use the score S_{uv}, which is bounded in $(\infty, -\infty)$, and derive from it a probability P_{uv}. To that end we project S_{uv} onto $(0, 1)$ using the sigmoid function, and define P_{uv} in the following manner:

$$P_{uv} = \frac{1}{1 + \exp(-S_{uv})} \tag{15}$$

Note that under this definition the log probability ratio is equal to the inverse of the sigmoid function:

$$\log \frac{P_{uv}}{1 - P_{uv}} = \log \frac{\frac{1}{1+\exp(-S_{uv})}}{\frac{\exp(-S_{uv})}{1+\exp(-S_{uv})}} = \log \frac{1}{\exp(-S_{uv})} = S_{uv} \tag{16}$$

Therefore, when we derive P_{uv} from S_{uv} with the sigmoid function, we can rewrite $\hat{G}_{prob}$ as:

$$\hat{G}_{prob} = \underset{G}{\text{argmax}} \sum_{u \neq v} S_{uv} \cdot I_{uv} + \log \eta \cdot |E| = \hat{G}_{score} \tag{17}$$

where we see that in this scenario the two objective functions are identical and the regularization term λ is related to the edge prior odds ratio by: $\lambda = -\log \eta$.

Moreover, assume that the score S_{uv} is computed as a linear combination over n features (such as a linear-kernel SVM), that is, $S_{uv} = \sum_{i=1}^{n} S_{uv}^{i} \cdot \alpha_i$, where S_{uv}^{i} denotes feature values and α_i denotes feature weights. In this case, the projected probability acquires the standard form of a logistic classifier:

$$P_{uv} = \frac{1}{1 + \exp(-\sum_{i=1}^{n} S_{uv}^{i} \cdot \alpha_i)} \tag{18}$$

Hence, we can train the weights α_i using a margin classifier and interpret the output of the classifier probabilistically, as we do with a logistic classifier. In our experiments in Section 5 we indeed use a linear-kernel SVM to train the weights α_i and then we can interchangeably interpret the resulting ILP as either score-based or probabilistic optimization.

4.2.4 Comparison to Snow, Jurafsky, and Ng (2006). Our work resembles Snow, Jurafsky, and Ng's work in that both try to learn graph edges given a transitivity constraint. There are two key differences in the model and in the optimization algorithm, however. First, they employ a greedy optimization algorithm that incrementally adds hyponyms to a large taxonomy (WordNet), whereas we simultaneously learn all edges using a global optimization method, which is more sound and powerful theoretically, and leads to the optimal solution. Second, Snow, Jurafsky, and Ng's model attempts to determine the graph that maximizes the likelihood $P(F|G)$ and not the posterior $P(G|F)$. If we cast their objective function as an ILP we get a formulation that is almost identical to ours, only containing the *inverse* prior odds ratio $\log \frac{1}{\eta} = -\log \eta$ rather than the prior odds ratio as the regularization term (cf. Section 2):

$$\hat{G}_{Snow} = \operatorname*{argmax}_{G} \sum_{u \neq v} \log \frac{P_{uv}}{(1 - P_{uv})} \cdot I_{uv} - \log \eta \cdot |E| \tag{19}$$

This difference is insignificant when $\eta \sim 1$, or when η is tuned empirically for optimal performance on a development set. If, however, η is statistically estimated, this might cause unwarranted results: Their model will favor dense graphs when the prior odds ratio is low ($\eta < 1$ or $P(I_{uv} = 1) < 0.5$), and sparse graphs when the prior odds ratio is high ($\eta > 1$ or $P(I_{uv} = 1) > 0.5$), which is counterintuitive. Our model does not suffer from this shortcoming because it optimizes the posterior rather than the likelihood. In Section 5 we show that our algorithm significantly outperforms the algorithm presented by Snow, Jurafsky, and Ng.

5. Experimental Evaluation

This section presents an evaluation and analysis of our algorithm.

5.1 Experimental Setting

A health-care corpus of 632MB was harvested from the Web and parsed using the Minipar parser (Lin 1998b). The corpus contains 2,307,585 sentences and almost 50 million

Table 2
The similarity score features used to represent pairs of templates. The columns specify the corpus over which the similarity score was computed, the template representation, the similarity measure employed, and the feature representation (as described in Section 4.1).

#	Corpus	Template	Similarity measure	Feature representation
1	health-care	binary	BInc	pair of CUI tuples
2	health-care	binary	BInc	pair of CUIs
3	health-care	binary	BInc	CUI tuple
4	health-care	binary	BInc	CUI
5	health-care	binary	Lin	pair of CUI tuples
6	health-care	binary	Lin	pair of CUIs
7	health-care	binary	Lin	CUI tuple
8	health-care	binary	Lin	CUI
9	health-care	unary	BInc	CUI tuple
10	health-care	unary	BInc	CUI
11	health-care	unary	Lin	CUI tuple
12	health-care	unary	Lin	CUI
13	RCV1	binary	Lin	lexical items
14	RCV1	unary	Lin	lexical items
15	RCV1	unary	BInc	lexical items
16	Lin & Pantel	binary	Lin	lexical items

word tokens. We used the Unified Medical Language System (UMLS)[6] to annotate medical concepts in the corpus. The UMLS is a database that maps natural language phrases to over one million **concept identifiers** in the health-care domain (termed **CUIs**). We annotated all nouns and noun phrases that are in the UMLS with their (possibly multiple) CUIs. We now provide the details of training an entailment classifier as explained in Section 4.1.

We extracted all templates from the corpus where both argument instantiations are medical concepts, that is, annotated with a CUI ($\sim$50,000 templates). This was done to increase the likelihood that the extracted templates are related to the health-care domain and reduce problems of ambiguity.

As explained in Section 4.1, a pair of templates constitutes an input example for the entailment classifier, and should be represented by a set of features. The features we used were different distributional similarity scores for the pair of templates, as summarized in Table 2. Twelve distributional similarity measures were computed over the health-care corpus using the aforementioned variations (Section 4.1), where two feature representations were considered: in the UMLS each natural language phrase may be mapped not to a *single* CUI, but to a *tuple* of CUIs. Therefore, in the first representation, each feature vector coordinate counts the number of times a tuple of CUIs was mapped to the term instantiating the template argument, and in the second representation it counts the number of times each single CUI was *one* of the CUIs mapped to the term instantiating the template argument. In addition, we obtained the original template similarity lists learned by Lin and Pantel (2001), and had available three distributional similarity measures learned by Szpektor and Dagan (2008), over the RCV1 corpus,[7] as detailed in Table 2. Thus, each pair of templates is represented by a total of 16 distributional similarity scores.

6 http://www.nlm.nih.gov/research/umls.
7 http://trec.nist.gov/data/reuters/reuters.html.

We automatically generated a balanced training set of 20,144 examples using WordNet and the procedure described in Section 4.1, and trained the entailment classifier with SVMperf (Joachims 2005). We use the trained classifier to obtain estimates for P_{uv} and S_{uv}, given that the score-based and probabilistic scoring functions are equivalent (cf. Section 4.2.3).

To evaluate the performance of our algorithm, we manually constructed gold-standard entailment graphs. First, 23 medical target concepts, representing typical topics of interest in the medical domain, were manually selected from a (longer) list of the most frequent concepts in the health-care corpus. The 23 target concepts are: *alcohol*, *asthma*, *biopsy*, *brain*, *cancer*, *CDC*, *chemotherapy*, *chest*, *cough*, *diarrhea*, *FDA*, *headache*, *HIV*, *HPV*, *lungs*, *mouth*, *muscle*, *nausea*, *OSHA*, *salmonella*, *seizure*, *smoking*, and *x-ray*. For each concept, we wish to learn a focused entailment graph (cf. Figure 1). Thus, the nodes of each graph were defined by extracting all propositional templates in which the corresponding target concept instantiated an argument at least $K(= 3)$ times in the health-care corpus (average number of graph nodes = 22.04, std = 3.66, max = 26, min = 13).

Ten medical students were given the nodes of each graph (propositional templates) and constructed the gold standard of graph edges using a Web interface. We gave an oral explanation of the annotation process to each student, and the first two graphs annotated by every student were considered part of the annotator training phase and were discarded. The annotators were able to select every propositional template and observe all of the instantiations of that template in our health-care corpus. For example, selecting the template *X helps with nausea* might show the propositions *relaxation helps with nausea*, *acupuncture helps with nausea*, and *Nabilone helps with nausea*. The concept of *entailment* was explained under the framework of TE (Dagan et al. 2009), that is, the template t_1 entails the template t_2 if given that the instantiation of t_1 with some concept is true then the instantiation of t_2 with the same concept is most likely true.

As explained in Section 3.2, we did not perform any disambiguation because a target concept disambiguates the propositional templates in focused entailment graphs. In practice, cases of ambiguity were very rare, except for a single scenario where in templates such as *X treats asthma*, annotators were unclear whether *X* is a type of doctor or a type of drug. The annotators were instructed in such cases to select the template, read the instantiations of the template in the corpus, and choose the sense that is most prevalent in the corpus. This instruction was applicable to all cases of ambiguity.

Each concept graph was annotated by two students. Following the current recognizing TE (RTE) practice (Bentivogli et al. 2009), after initial annotation the two students met for a reconciliation phase. They worked to reach an agreement on differences and corrected their graphs. Inter-annotator agreement was calculated using the kappa statistic (Siegel and Castellan 1988) both before ($\kappa = 0.59$) and after ($\kappa = 0.9$) reconciliation. Each learned graph was evaluated against the two reconciliated graphs.

Summing the number of possible edges over all 23 concept graphs we get 10,364 possible edges, of which 882 on average were included by the annotators (averaging over the two gold-standard annotations for each graph). The concept graphs were randomly split into a development set (11 concepts) and a test set (12 concepts).

We used the *lpsolve*[8] package to learn the edges of the graphs. This package efficiently solves the model without imposing integer restrictions[9] and then uses the branch-and-bound method to find an optimal integer solution. We note that in the

8 `http://lpsolve.sourceforge.net/5.5/`.

9 While ILP is an NP-hard problem, LP is a polynomial problem and can be solved efficiently.

experiments reported in this article the optimal solution without integer restrictions was already integer. Thus, although in general our optimization problem is NP-hard, in our experiments we were able to reach an optimal solution for the input graphs very efficiently (we note that in some scenarios not reported in this article the optimal solution was not integer and so an integer solution is not guaranteed a priori).

As mentioned in Section 4.2, we added a few constraints in cases where there was strong evidence that edges are not in the graph. This is done in the following scenarios (examples given in Table 3): (1) When two templates u and v are identical except for a pair of words w_u and w_v, and w_u is an antonym of w_v, or a hypernym of w_v at distance ≥ 2 in WordNet. (2) When two nodes u and v are transitive "opposites," that is, if $u = X \xleftarrow{subj} w \xrightarrow{obj} Y$ and $v = X \xleftarrow{obj} w \xrightarrow{subj} Y$, for any word w. We note that there are some transitive verbs that express a reciprocal activity, such as *X marries Y*, but usually reciprocal events are not expressed using a transitive verb structure.

In addition, in some cases we have strong evidence that edges do exist in the graph. This is done in a single scenario (see Table 3), which is specific to the output of Minipar: when two templates differ by a single edge and the first is of the type $X \xrightarrow{obj} Y$ and the other is of the type $X \xleftarrow{vrel} Y$, which expresses a passive verb modifier of nouns. Altogether, these initializations took place in less than 1% of the node pairs in the graphs. We note that we tried to use WordNet relations such as hypernym and synonym as "positive" hard constraints (using the constraint $I_{uv} = 1$), but this resulted in reduced performance because the precision of WordNet was not high enough.

The graphs learned by our algorithm were evaluated by two measures. The first measure evaluates the graph edges directly, and the second measure is motivated by semantic inference applications that utilize the rules in the graph. The first measure is simply the F_1 of the set of learned edges compared to the set of gold-standard edges. In the second measure we take the set of learned rules and infer new propositions by applying the rules over all propositions extracted from the health-care corpus. We apply the rules iteratively over all propositions until no new propositions are inferred. For example, given the corpus proposition *relaxation reduces nausea* and the edges *X reduces nausea* $\rightarrow$ *X helps with nausea* and *X helps with nausea* $\rightarrow$ *X related to nausea*, we evaluate the set {*relaxation reduces nausea, relaxation helps with nausea, relaxation related to nausea*}. For each graph we measure the F_1 of the set of propositions inferred by the learned graphs when compared to the set of propositions inferred by the gold-standard graphs. For both measures the final score of an algorithm is a macro-average F_1 over the 24 gold-standard test-set graphs (two gold-standard graphs for each of the 12 test concepts).

Table 3
Scenarios in which we added hard constraints to the ILP.

Scenario	Example	Initialization
antonym	($X \xleftarrow{subj}$ *decrease* $\xrightarrow{obj} Y$,$X \xleftarrow{subj}$ *increase* $\xrightarrow{obj} Y$)	$I_{uv} = 0$
hypernym ≥ 2	($X \xleftarrow{subj}$ *affect* $\xrightarrow{obj} Y$,$X \xleftarrow{subj}$ *irritate* $\xrightarrow{obj} Y$)	$I_{uv} = 0$
transitive opposite	($X \xleftarrow{subj}$ *cause* $\xrightarrow{obj} Y$,$Y \xleftarrow{subj}$ *cause* $\xrightarrow{obj} X$)	$I_{uv} = 0$
syntactic variation	($X \xleftarrow{subj}$ *follow* $\xrightarrow{obj} Y$,$Y \xleftarrow{subj}$ *follow* $\xleftarrow{vrel} X$)	$I_{uv} = 1$

Learning the edges of a graph given an input concept takes about 1–2 seconds on a standard desktop.

5.2 Evaluated Algorithms

First, we describe some baselines that do not utilize the entailment classifier or the ILP solver. For each of the 16 distributional similarity measures (Table 2) and for each template t, we computed a list of templates most similar to t (or entailing t for directional measures). Then, for each measure we learned graphs by inserting an edge (u, v), when u is in the top K templates most similar to v. The parameter K can be optimized either on the automatically generated training set (from WordNet) or on the manually annotated development set. We also learned graphs using WordNet: We inserted an edge (u, v) when u and v differ by a single word w_u and w_v, respectively, and w_u is a direct hyponym or synonym of w_v. Next, we describe algorithms that utilize the entailment classifier.

Our algorithm, named **ILP-Global**, utilizes global information and an ILP formulation to find maximum a posteriori graphs. Therefore, we compare it to the following three variants: (1) **ILP-Local**: An algorithm that uses only local information. This is done by omitting the global transitivity constraints, and results in an algorithm that inserts an edge (u, v) if and only if $(S_{uv} - \lambda) > 0$. (2) **Greedy-Global**: An algorithm that looks for the maximum a posteriori graphs but only employs the greedy optimization procedure as described by Snow, Jurafsky, and Ng (2006). (3) **ILP-Global-Likelihood**: An ILP formulation where we look for the maximum likelihood graphs, as described by Snow, Jurafsky, and Ng (cf. Section 4.2).

We evaluate these algorithms in three settings which differ in the method by which the edge prior odds ratio, η (or λ), is estimated: (1) $\eta = 1$ ($\lambda = 0$), which means that no prior is used. (2) Tuning η and using the value that maximizes performance over the development set. (3) Estimating η using maximum likelihood over the development set, which results in $\eta \sim 0.1$ ($\lambda \sim 2.3$), corresponding to the edge density $P(I_{uv} = 1) \sim 0.09$.

For all local algorithms whose output does not respect transitivity constraints, we added all edges inferred by transitivity. This was done because we assume that the rules learned are to be used in the context of an inference or entailment system. Because such systems usually perform chaining of entailment rules (Raina, Ng, and Manning 2005; Bar-Haim et al. 2007; Harmeling 2009), we conduct this chaining as well. Nevertheless, we also measured performance when edges inferred by transitivity are not added: We once again chose the edge prior value that maximizes F_1 over the development set and obtained macro-average recall/precision/F_1 of 51.5/34.9/38.3. This performance is comparable to the macro-average recall/precision/F_1 of 44.5/45.3/38.1 we report next in Table 4.

5.3 Experimental Results and Analysis

In this section we present experimental results and analysis that show that the ILP-Global algorithm improves performance over baselines, specifically in terms of precision.

Tables 4–7 and Figure 2 summarize the performance of the algorithms. Table 4 shows our main result when the parameters λ and K are optimized to maximize performance over the development set. Notice that the algorithm ILP-Global-Likelihood is omitted, because when optimizing λ over the development set it conflates with ILP-Global. The rows $\textbf{Local}_1$ and $\textbf{Local}_2$ present the best algorithms that use a single distributional similarity resource. Local_1 and Local_2 correspond to the configurations

Table 4
Results when tuning for performance over the development set.

	Edges			Propositions		
	Recall	Precision	F_1	Recall	Precision	F_1
ILP-Global ($\lambda = 0.45$)	46.0	50.1	**43.8**	67.3	69.6	**66.2**
Greedy-Global ($\lambda = 0.3$)	45.7	37.1	36.6	64.2	57.2	56.3
ILP-Local ($\lambda = 1.5$)	44.5	45.3	38.1	65.2	61.0	58.6
Local_1 ($K = 10$)	53.5	34.9	37.5	73.5	50.6	56.1
Local_2 ($K = 55$)	52.5	31.6	37.7	69.8	50.0	57.1

Table 5
Results when the development set is not used to estimate λ and K.

	Edges			Propositions		
	Recall	Precision	F_1	Recall	Precision	F_1
ILP-Global	58.0	28.5	**35.9**	76.0	46.0	**54.6**
Greedy-Global	60.8	25.6	33.5	77.8	41.3	50.9
ILP-Local	69.3	19.7	26.8	82.7	33.3	42.6
Local_1 ($K = 100$)	92.6	11.3	20.0	95.3	18.9	31.1
Local_2 ($K = 100$)	63.1	25.5	34.0	77.7	39.9	50.9
WordNet	10.8	44.1	13.2	39.9	72.4	47.3

Table 6
Results with prior estimated on the development set, that is $\eta = 0.1$, which is equivalent to $\lambda = 2.3$.

	Edges			Propositions		
	Recall	Precision	F_1	Recall	Precision	F_1
ILP-Global	16.8	67.1	**24.4**	43.9	86.8	**56.3**
ILP-Global-Likelihood	91.8	9.8	17.5	94.0	16.7	28.0
Greedy-Global	14.7	62.9	21.2	43.5	86.6	56.2
Greedy-Global-Likelihood	100.0	9.3	16.8	100.0	15.5	26.5

described in Table 2 by features no. 5 and no. 1, respectively (see also Table 8). ILP-Global improves performance by at least 13%, and significantly outperforms all local methods, as well as the greedy optimization algorithm both on the edges F_1 measure ($p < 0.05$) and on the propositions F_1 measure ($p < 0.01$).[10]

Table 5 describes the results when the development set is not used to estimate the parameters λ and K: A uniform prior ($P_{uv} = 0.5$) is assumed for algorithms that use the entailment classifier, and the automatically generated training set is employed to estimate K. Again ILP-Global-Likelihood is omitted in the absence of a prior. ILP-Global outperforms all other methods in this scenario as well, although by a smaller margin for a few of the baselines. Comparing Table 4 to Table 5 reveals that excluding the

10 We tested significance using the two-sided Wilcoxon rank test (Wilcoxon 1945).

Table 7
Results per concept for the ILP-Global.

Concept	R	P	F_1
Smoking	58.1	81.8	67.9
Seizure	64.7	51.2	57.1
Headache	60.9	50.0	54.9
Lungs	50.0	56.5	53.1
Diarrhea	42.1	60.0	49.5
Chemotherapy	44.7	52.5	48.3
HPV	35.2	76.0	48.1
Salmonella	27.3	80.0	40.7
X-ray	75.0	23.1	35.3
Asthma	23.1	30.6	26.3
Mouth	17.7	35.5	23.7
FDA	53.3	15.1	23.5

sparse prior indeed increases recall at a price of a sharp decrease in precision. Note, however, that local algorithms are more vulnerable to this phenomenon. This makes sense because in local algorithms eliminating the prior adds edges that in turn add more edges due to the constraint of transitivity and so recall dramatically rises at the expense of precision. Global algorithms are not as prone to this effect because they refrain from adding edges that eventually lead to the addition of many unwarranted edges.

Table 5 also shows that WordNet, a manually constructed resource, has notably the highest precision and lowest recall. The low recall exemplifies how the entailment relations given by the gold-standard annotators transcend much beyond simple lexical relations that appear in WordNet: Many of the gold-standard entailment relations are missing from WordNet or involve multi-word phrases that do not appear in WordNet at all.

Note that although the precision of WordNet is the highest in Table 5, its absolute value (44.1%) is far from perfect. This illustrates that hierarchies of predicates are quite

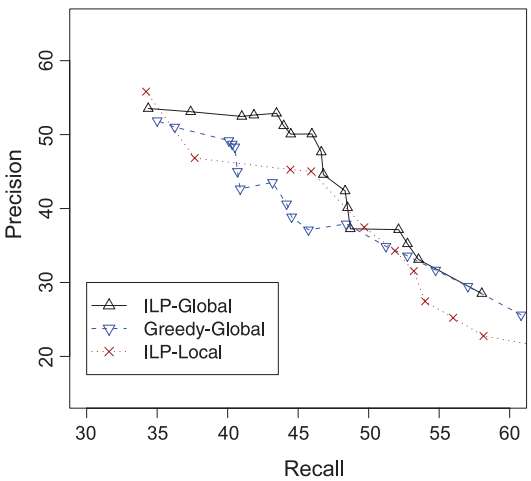

Figure 2
Recall-precision curve comparing ILP-Global with Greedy-Global and ILP-Local.

Table 8
Results of all distributional similarity measures when tuning K over the development set. We encode the description of the measures presented in Table 2 in the following manner—*h = health-care corpus*; *R = RCV1 corpus*; *b = binary templates*; *u = unary templates*; *L = Lin similarity measure*; *B = BInc similarity measure*; *pCt = pair of CUI tuples representation*; *pC = pair of CUIs representation*; *Ct = CUI tuple representation*; *C = CUI representation*; *Lin & Pantel = similarity lists learned by Lin and Pantel*.

	Edges			Propositions		
Dist. sim. measure	Recall	Precision	F_1	Recall	Precision	F_1
h-b-B-pCt	52.5	31.6	**37.7**	69.8	50.0	**57.1**
h-b-B-pC	50.5	26.5	30.7	67.1	43.5	50.1
h-b-B-Ct	10.4	44.5	15.4	39.1	78.9	51.6
h-b-B-C	7.6	42.9	11.1	37.9	79.8	50.7
h-b-L-pCt	53.4	34.9	37.5	73.5	50.6	56.1
h-b-L-pC	47.2	35.2	35.6	68.6	52.9	56.2
h-b-L-Ct	47.0	26.6	30.2	64.9	47.4	49.6
h-b-L-C	34.6	22.9	22.5	57.2	52.6	47.6
h-u-B-Ct	5.1	37.4	8.5	35.1	91.0	49.7
h-u-B-C	7.2	42.4	11.5	36.1	90.3	50.1
h-u-L-Ct	22.8	22.0	18.3	49.7	49.2	44.5
h-u-L-C	16.7	26.3	17.8	47.0	56.8	48.1
R-b-L-l	49.4	21.8	25.2	72.4	39.0	45.5
R-u-L-l	24.1	30.0	16.8	47.1	55.2	42.1
R-u-B-l	9.5	57.1	14.1	37.2	84.0	49.5
Lin & Pantel	37.1	32.2	25.1	58.9	54.6	48.6

ambiguous and thus using WordNet directly yields relatively low precision. WordNet is vulnerable to such ambiguity because it is a generic domain-independent resource, whereas our algorithm learns from a domain-specific corpus. For example, the words *have* and *cause* are synonyms according to one of the senses in WordNet and so the erroneous rule *X have asthma* $\leftrightarrow$ *X cause asthma* is learned using WordNet. Another example is the rule *X follows chemotherapy* $\rightarrow$ *X takes chemotherapy*, which is incorrectly inferred because *follow* is a hyponym of *take* according to one of WordNet's senses (*she followed the feminist movement*). Due to these mistakes made by WordNet, the precision achieved by our automatically trained ILP-Global algorithm when tuning parameters on the development set (Table 4) is higher than that of WordNet.

Table 6 shows the results when the prior η is estimated using maximum likelihood over the development set (by computing the edge density over all the development set graphs), and not tuned empirically with grid search. This allows for a comparison between our algorithm that maximizes the a posteriori probability and Snow, Jurafsky, and Ng's (2006) algorithm that maximizes the likelihood. The gold-standard graphs are quite sparse ($\eta \sim 0.1$); therefore, as explained in Section 4.2.4, the effect of the prior is substantial. ILP-Global and Greedy-Global learn sparse graphs with high precision and low recall, whereas ILP-Global-Likelihood and Greedy-Global-Likelihood learn dense graphs with high recall but very low precision. Overall, optimizing the a posteriori probability is substantially better than optimizing likelihood, but still leads to a large degradation in performance. This can be explained because our algorithm is not purely probabilistic: The learned graphs are the product of mixing a probabilistic objective function with non-probabilistic constraints. Thus, plugging the estimated prior into this model results in performance that is far from optimal. In future work, we will examine

a purely probabilistic approach that will allow us to reach good performance when estimating η directly. Nevertheless, currently optimal results are achieved when the prior η is tuned empirically.

Figure 2 shows a recall–precision curve for ILP-Global, Greedy-Global, and ILP-Local, obtained by varying the prior parameter, λ. The figure clearly demonstrates the advantage of using global information and ILP. ILP-Global is better than Greedy-Global and ILP-Local in almost every point of the recall–precision curve, regardless of the exact value of the prior parameter. Last, we present for completeness in Table 7 the results of ILP-Global for all concepts in the test set.

In Table 8 we present the results obtained for all 16 distributional similarity measures. The main conclusion we can derive from this table is that the best distributional similarity measures are those that represent templates using *pairs* of argument instantiations rather than each argument separately. A similar result was found by Yates and Etzioni (2009), who described the RESOLVER paraphrase learning system and have shown that it outperforms DIRT. In their analysis, they attribute this result to their representation that utilizes pairs of arguments comparing to DIRT, which computes a separate score for each argument.

In the next two sections we perform a more thorough qualitative and quantitative comparison trying to analyze the importance of using global information in graph learning (Section 5.3.1), as well as the contribution of using ILP rather than a greedy optimization procedure (Section 5.3.2). We note that the analysis presented in both sections is for the results obtained when optimizing parameters over the development set.

5.3.1 Global vs. Local Information. We looked at all edges in the test-set graphs where ILP-Global and ILP-Local disagree and checked which algorithm was correct. Table 9 presents the result. The main advantage of using ILP-Global is that it avoids inserting wrong edges into the graph. This is because ILP-Local adds any edge (u, v) such that P_{uv} crosses a certain threshold, disregarding edges that will be consequently added due to transitivity (recall that for local algorithms we add edges inferred by transitivity, cf. Section 5.2). ILP-Global will avoid such edges of high probability if it results in inserting many low probability edges. This results in an improvement in precision, as exhibited by Table 4.

Figures 3 and 4 show fragments of the graphs learned by ILP-Global and ILP-Local (prior to adding transitive edges) for the test-set concepts *diarrhea* and *seizure*, and illustrate qualitatively how global considerations improve precision. In Figure 3, we witness that the single erroneous edge *X results in diarrhea* $\rightarrow$ *X prevents diarrhea* inserted by the local algorithm because P_{uv} is high, effectively bridges two strongly connected components and induces a total of 12 wrong edges (all edges from the upper component to the lower component), whereas ILP-Global refrains from inserting this edge. Figure 4 depicts an even more complex scenario. First, ILP-Local induces a strongly connected component of five nodes for the predicates *control*, *treat*, *stop*,

Table 9
Comparing disagreements between ILP-Global and ILP-Local against the gold-standard graphs.

	Global=True/Local=False	Global=False/Local=True
Gold standard=true	48	42
Gold standard=false	78	494

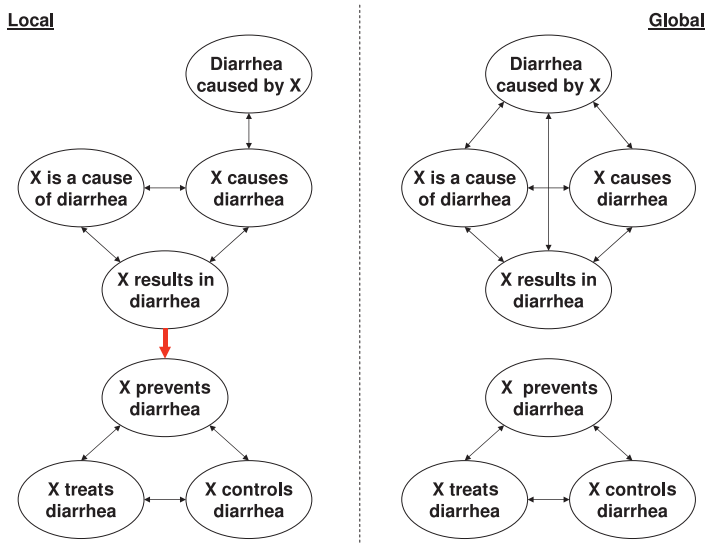

Figure 3
A comparison between ILP-Global and ILP-local for two fragments of the test-set concept *diarrhea*.

reduce, and *prevent*, whereas ILP-Global splits this strongly connected component into two, which although not perfect, is more compatible with the gold-standard graphs. In addition, ILP-Local inserts four erroneous edges that connect two components of size 4 and 5, which results in adding eventually 30 wrong edges. On the other hand,

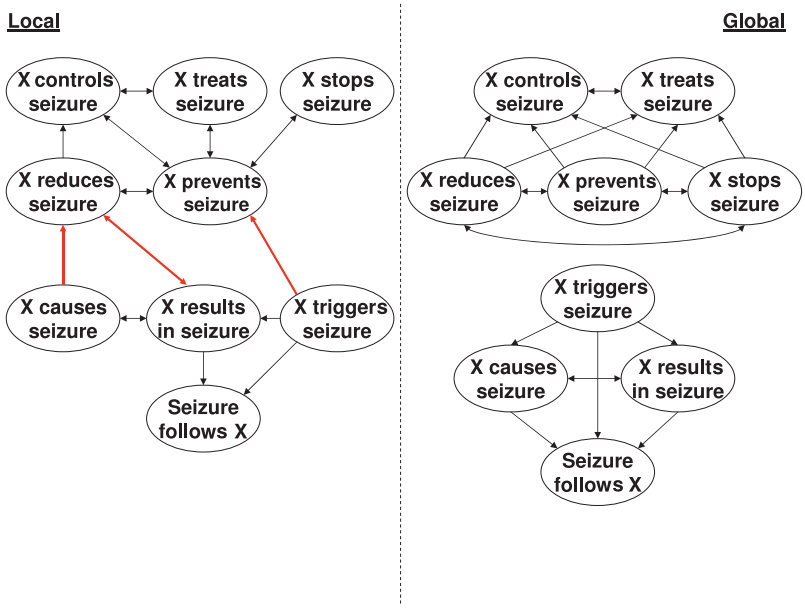

Figure 4
A comparison between ILP-Global and ILP-Local for two fragments of the test-set concept *seizure*.

ILP-Global is aware of the consequences of adding these four seemingly good edges, and prefers to omit them from the learned graph, leading to much higher precision.

Although the main contribution of ILP-Global, in terms of F_1, is in an increase in precision, we also notice an increase in recall in Table 4. This is because the optimal prior is $\lambda = 0.45$ in ILP-Global but $\lambda = 1.5$ in ILP-Local. Thus, any edge (u, v) such that $0.45 < S_{uv} < 1.5$ will have positive weight in ILP-Global and might be inserted into the graph, but will have negative weight in ILP-Local and will be rejected. The reason is that in a local setting, reducing false positives is handled only by applying a large penalty for every wrong edge, whereas in a global setting wrong edges can be rejected because they induce more "bad" edges. Overall, this leads to an improved recall in ILP-Global. This also explains why ILP-Local is severely harmed when no prior is used at all, as shown in Table 5.

Last, we note that across the 12 test-set graphs, ILP-Global achieves better F_1 over the edges in 7 graphs with an average advantage of 11.7 points, ILP-Local achieves better F_1 over the edges in 4 graphs with an average advantage of 3.0 points, and one performance is equal.

5.3.2 Greedy vs. Non-Greedy Optimization. We would like to understand how using an ILP solver improves performance compared with a greedy optimization procedure. Table 4 demonstrates that ILP-Global and Greedy-Global reach a similar level of recall, although ILP-Global achieves far better precision. Again, we investigated edges for which the two algorithms disagree and checked which one was correct. Table 10 demonstrates that the higher precision is because ILP-Global avoids inserting wrong edges into the graph.

Figure 5 illustrates some of the reasons ILP-Global performs better than Greedy-Global. Parts A1–A3 show the progression of Greedy-Global, which is an incremental algorithm, for a fragment of the *headache* graph. In part A1 the learning algorithm still separates the nodes *X prevents headache* and *X reduces headache* from the nodes *X causes headache* and *X results in headache* (nodes surrounded by a bold oval shape constitute a strongly connected component). After two iterations, however, the four nodes are joined into a single strongly connected component, which is an error in principle but at this point seems to be the best decision to increase the posterior probability of the graph. This greedy decision has two negative ramifications. First, the strongly connected component can no longer be untied. Thus, in A3 we observe that in future iterations the strongly connected component expands further and many more wrong edges are inserted into the graph. On the other hand, in B we see that ILP-Global takes into consideration the global interaction between the four nodes and other nodes of the graph, and decides to split this strongly connected component in two, which improves the precision of ILP-Global. Second, note that in A3 the nodes *Associate X with headache* and *Associate headache with X* are erroneously isolated. This is because connecting them to the strongly connected component that contains six nodes will add many edges with

Table 10
Comparing disagreements between ILP-Global and Greedy-Global against the gold-standard graphs.

	ILP=True/Greedy=False	ILP=False/Greedy=True
Gold standard=true	66	56
Gold standard=false	44	480

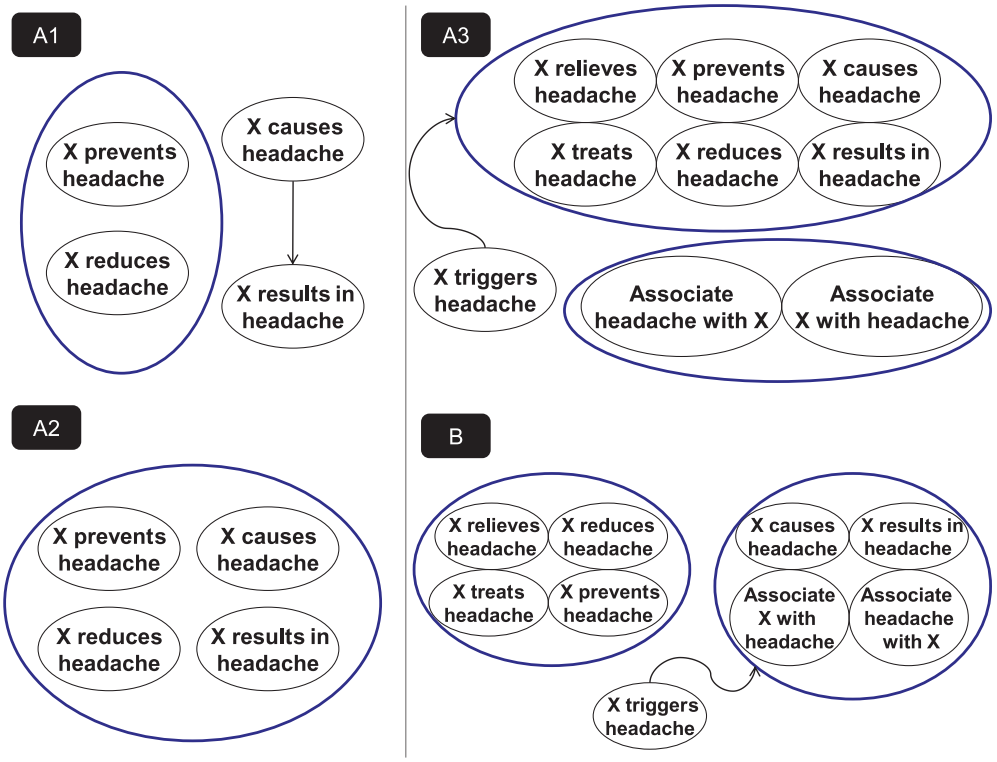

Figure 5
A comparison between ILP-Global and Greedy-Global. Parts A1–A3 depict the incremental progress of Greedy Global for a fragment of the *headache* graph. Part B depicts the corresponding fragment in ILP-Global. Nodes surrounded by a bold oval shape are strongly connected components.

low probability and so this is avoided by Greedy-Global. Because in ILP-Global the strongly connected component was split in two, it is possible to connect these two nodes to some of the other nodes and raise the recall of ILP-Global. Thus, we see that greedy optimization might get stuck in local maxima and consequently suffer in terms of both precision and recall.

Last, we note that across the 12 test-set graphs, ILP-Global achieves better F_1 over the edges in 9 graphs with an average advantage of 10.0 points, Greedy-Global achieves better F_1 over the edges in 2 graphs with an average advantage of 1.5 points, and in one case performance is equal.

5.4 Error Analysis

In this section, we compare the results of ILP-Global with the gold-standard graphs and perform error analysis. Error analysis was performed by comparing the 12 graphs learned by ILP-Global to the corresponding 12 gold-standard graphs (randomly sampling from the two available gold-standard graphs), and manually examining all edges for which the two disagree. We found that the number of false positives and false negatives is almost equal: 282 edges were learned by ILP-Global but are not in the gold-standard graphs (false positive) and 287 edges were in the gold-standard graphs but were not learned by ILP-Global (false negatives).

Table 11
Error analysis for false positives and false negatives.

False positives		False negatives	
Total count	282	Total count	287
Classifier error	84.8%	Classifier error	73.5%
Co-hyponym error	18.0%	Long-predicate error	36.2%
Direction error	15.1%	Generality error	26.8%
		String overlap error	20.9%

Table 11 presents the results of our manual error analysis. Most evident is the fact that the majority of mistakes are misclassifications of the entailment classifier. For 73.5% of the false negatives the classifier's probability was $P_{uv} < 0.5$ and for 84.8% of the false positives the classifier's probability was $P_{uv} > 0.5$. This shows that our current classifier struggles to distinguish between positive and negative examples. Figure 6 illustrates some of this difficulty by showing the distribution of the classifier's probability, P_{uv}, over all node pairs in the 12 test-set graphs. Close to 80% of the scores are in the range 0.45–0.5, most of which are simply node pairs for which all distributional similarity features are zero. Although in the great majority of such node pairs (t_1, t_2) t_1 indeed does not entail t_2, there are also some cases where t_1 does entail t_2. This implies that the current feature representation is not rich enough, and in the next section we explore a larger feature set.

Table 11 also shows some other reasons found for false positives. Many false positives are pairs of predicates that are semantically related, that is, 18% of false positives are templates that are hyponyms of a common predicate (co-hyponym error), and 15.1% of false positives are pairs where we err in the direction of entailment (direction error). For example ILP-Global learns that *place X in mouth* → *remove X from mouth*, which is a

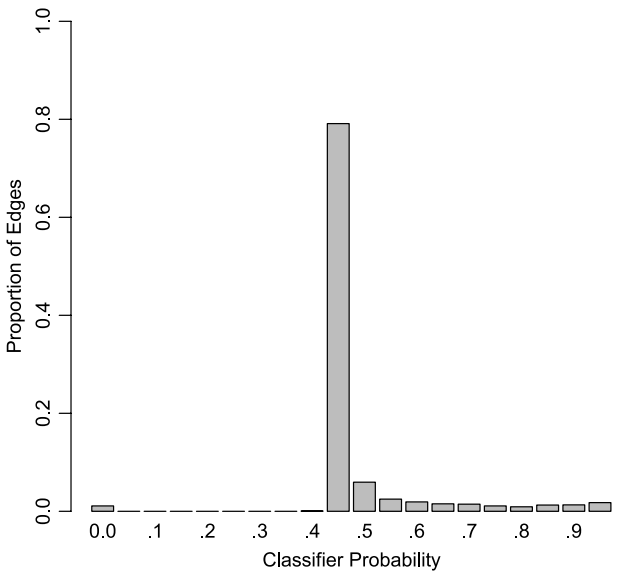

Figure 6
Distribution of probabilities given by the classifier over all node pairs of the test-set graphs.

co-hyponym error, and also that *X affects lungs* → *X damages lungs*, which is a direction error because entailment holds in the other direction. This illustrates the infamous difficulty of distributional similarity features to discern the type of semantic relation between two predicates.

Table 11 also shows additional reasons for false negatives. We found that in 36.2% of false negatives one of the two templates contained a "long" predicate, that is a predicate composed of more than one content word, such as *Ingestion of X causes injury to Y*. This might indicate that the size of the health-care corpus is too small to collect sufficient statistics for complex predicates. In addition, 26.8% of false negatives were manually analyzed as "generality errors." An example is the edge *HPV strain causes X* → *associate HPV with X* that is in the gold-standard graph but was missed by ILP-Global. Indeed, this edge falls within the definition of textual entailment and is correct: For example, if some HPV strain causes cervical cancer then cervical cancer is associated with HPV. Because the entailed template is much more general than the entailing template, however, they are not instantiated by similar arguments in the corpus and distributional similarity features fail to capture their semantic similarity. Last, we note that in 20.9% of the false negatives, there was some string overlap between the entailing and entailed templates, for example in *X controls asthma symptoms* → *X controls asthma*. In the next section we experiment with a feature that is based on string similarity.

Tables 8 and 9 show that there are cases where ILP-Global makes a mistake, whereas ILP-Local or Greedy-Global are correct. An illustrating example for such a case is shown in Figure 7. Looking at ILP-Local we see that the entailment classifier correctly classifies the edges *X triggers asthma* → *X causes asthma* and *X causes asthma* → *Associate X with asthma*, but misclassifies *X triggers asthma* → *Associate X with asthma*. Because this configuration violates a transitivity constraint, ILP-Global must make a global decision whether to add the edge *X triggers asthma* → *Associate X with asthma* or to omit one of

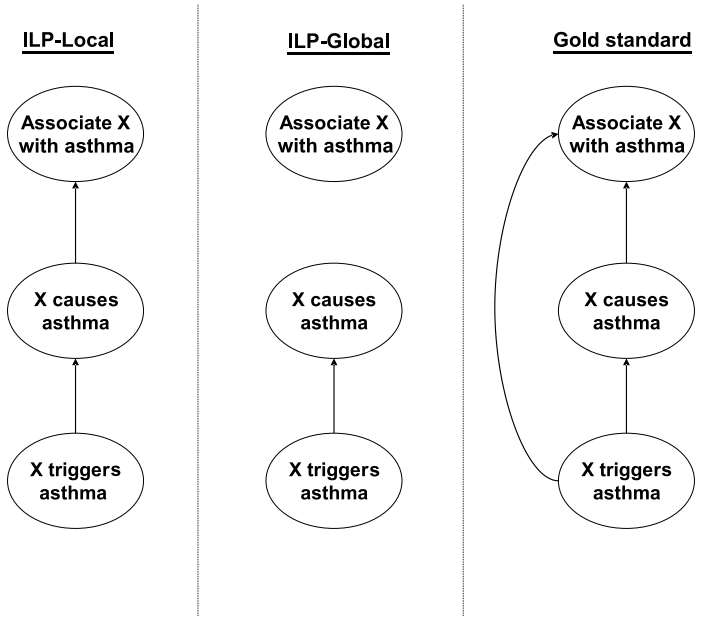

Figure 7
A scenario where ILP-Global makes a mistake, but ILP-Local is correct.

the correct edges. The optimal global decision in this case causes a mistake with respect to the gold standard. More generally, a common phenomenon of ILP-Global is that it splits components that are connected in ILP-Local, for example, in Figures 3 and 4. ILP-Global splits the components in a way that is optimal according to the scores of the local entailment classifier, but these are not always accurate according to the gold standard.

Figure 5 exemplifies a scenario where ILP-Global errs, but Greedy-Global is (partly) correct. ILP-Global mistakenly learns entailment rules from the templates *Associate X with headache* and *Associate headache with X* to the templates *X causes headache* and *X results in headache*, whereas Greedy-Global isolates the templates *Associate X with headache* and *Associate headache with X* in a separate component. This happens because of the greedy nature of Greedy-Global. Notice that in step A2 the templates *X causes headache* and *X results in headache* are already included (erroneously) in a connected component with the templates *X prevents headache* and *X reduces headache*. Thus, adding the rules from *Associate X with headache* and *Associate headache with X* to *X causes headache* and *X results in headache* would also add the rules to *X reduces headache* and *X prevents headache* and the Greedy-Global avoids that. ILP-Global does not have that problem: It simply chooses the optimal choice according to the entailment classifier, which splits the connected component presented in A2. Thus, once again we see that mistakes made by ILP-Global are often due to the inaccuracies of the scores given by the local entailment classifier.

6. Local Classifier Extensions

The error analysis in Section 5.4 exemplified that most errors are the result of misclassifications made by the local entailment classifier. In this section, we investigate the local entailment classifier component, focusing on the set of features used for classification. We first present an experimental setting in which we consider a wider set of features, then we present the results of the experiment, and last we perform feature analysis and draw conclusions.

6.1 Feature Set and Experimental Setting

In previous sections we employed a distant supervision framework: We generated training examples automatically with WordNet, and represented each example with distributional similarity features. Distant supervision comes with a price, however—it prevents us from utilizing all sources of information. For example, looking at the pair of gold-standard templates *X manages asthma* and *X improves asthma management*, one can exploit the fact that *management* is a derivation of *manage* to improve the estimation of entailment. The automatically generated training set was generated by looking at WordNet's hypernym, synonym, and co-hyponyms relations, however, and hence no such examples appear in the training set, rendering this type of feature useless. Moreover, one cannot use WordNet's hypernym, synonym, and co-hyponym relations as features because the generated training set is highly biased—all positive training examples are either hypernyms or synonyms and all negative examples are co-hyponyms.

In this section we would like to examine the utility of various features, while avoiding the biases that occur due to distant supervision. Therefore, we use the 23 manually annotated gold-standard graphs for both training and testing, in a cross-validation setting. Although this reduces the size of the training set it allows us to estimate the utility of various features in a setting where the training set and test set are sampled from the same underlying distribution, without the aforementioned biases.

We would like to extract features that express information that is diverse and orthogonal to the one given by distributional similarity. Therefore, we turn to existing knowledge resources that were created using both manual and automatic methods, expressing various types of linguistic and statistical information that is relevant for entailment prediction:

1. WordNet: contains manually annotated relations such as hypernymy, synonymy, antonymy, derivation, and entailment.
2. VerbOcean[11] (Chklovski and Pantel 2004): contains verb relations such as *stronger-than* and *similar* that were learned with pattern-based methods.
3. CATVAR[12] (Habash and Dorr 2003): contains word derivations such as *develop–development*.
4. FRED[13] (Ben Aharon, Szpektor, and Dagan 2010): contains entailment rules between templates learned automatically from FrameNet.
5. NomLex[14] (Macleod et al. 1998): contains English nominalizations including their argument mapping to the corresponding verbal form.
6. BAP[15] (Kotlerman et al. 2010): contains directional distributional similarity scores between lexical terms (rather than propositional templates) calculated with the BAP similarity scoring function.

Table 12 describes the 16 new features that were generated for each of the gold-standard examples (resulting in a total of 32 features). The first 15 features were generated by the aforementioned knowledge bases. The last feature measures the edit distance between templates: Given a pair of templates (t_1, t_2), we concatenate the words in each template and derive a pair of strings (s_1, s_2). Then we compute the Levenshtein string edit-distance (Cohen, Ravikumar, and Fienberg 2003) between s_1 and s_2 and divide the score by $|s_1| + |s_2|$ for normalization.

Table 12 also describes for each feature the number and percentage of examples for which the feature value is non-zero (out of the examples generated from the 23 gold-standard graphs). A salient property of many of the new features is that they are sparse: The four antonymy features as well as the Derivation, Entailment, Nomlex, and FRED features occur in very few examples in our data set, which might make training with these features difficult.

After generating the new features we employ a leave-one-graph-out strategy to maximally exploit the manually annotated gold standard for training. For each of the *test-set* graphs, we train over all development and test-set graphs except for the one that is left out,[16] after tuning the algorithm's parameters and test. Parameter tuning is done by cross-validation over the development set, tuning to maximize the F_1 of the set

11 http://demo.patrickpantel.com/demos/verbocean/.
12 http://clipdemos.umiacs.umd.edu/catvar/.
13 http://u.cs.biu.ac.il/∼nlp/downloads/FRED.html.
14 http://nlp.cs.nyu.edu/nomlex/index.html.
15 http://u.cs.biu.ac.il/∼nlp/downloads/DIRECT.html.
16 As described in Section 5, we train with a balanced number of positive and negative examples. Because the number of positive examples in the gold standard is smaller than the number of negative examples, we use all positives and randomly sample the same number of negatives, resulting in $\sim 1,500$ training examples.

Table 12
The set of new features. The last two columns denote the number and percentage of examples for which the value of the feature is non-zero in examples generated from the 23 gold-standard graphs.

Name	Type	Description	#	%
Hyper.	boolean	Whether (t_1, t_2) are identical except for a pair of words (w_1, w_2) such that w_2 is a hypernym (distance ≤ 2) of w_1 in WordNet.	120	1.1
Syno.	boolean	Whether (t_1, t_2) are identical except for a pair of words (w_1, w_2) such that w_2 is a synonym of w_1 in WordNet.	94	0.9
Co-hypo.	boolean	Whether (t_1, t_2) are identical except for a pair of words (w_1, w_2) such that w_2 is a co-hyponym of w_1 in WordNet.	302	2.8
WN Ant.	boolean	Whether (t_1, t_2) are identical except for a pair of words (w_1, w_2) such that w_2 is an antonym of w_1 in WordNet.	6	0.06
VO Ant.	boolean	Whether (t_1, t_2) are identical except for a pair of words (w_1, w_2) such that w_2 is an antonym of w_1 in VerbOcean.	25	0.2
WN Ant. 2	boolean	Whether there exists in (t_1, t_2) a pair of words (w_1, w_2) such that w_2 is an antonym of w_1 in WordNet.	22	0.2
VO Ant. 2	boolean	Whether there exists in (t_1, t_2) a pair of words (w_1, w_2) such that w_2 is an antonym of w_1 in VerbOcean.	73	0.7
Derivation	boolean	Whether there exists in (t_1, t_2) a pair of words (w_1, w_2) such that w_2 is a derivation of w_1 in WordNet or CATVAR.	78	0.7
Entailment	boolean	Whether there exists in (t_1, t_2) a pair of words (w_1, w_2) such that w_2 is entailed by w_1 in WordNet.	20	0.2
FRED	boolean	Whether t_1 entails t_2 in FRED.	9	0.08
Nomlex	boolean	Whether t_1 entails t_2 in Nomlex.	8	0.07
VO strong	boolean	Whether (t_1, t_2) are identical except for a pair of words (w_1, w_2) such that w_2 is *stronger than* w_1 in VerbOcean.	104	1
VO simil.	boolean	Whether (t_1, t_2) are identical except for a pair of words (w_1, w_2) such that w_2 is *similar to* w_1 in VerbOcean.	191	1.8
Positive	boolean	Disjunction of the features Hypernym, Synonym, Nomlex, and VO stronger.	289	2.7
BAP	real	$max_{w_1 \in t_1, w_2 \in t_2} BAP(w_1, w_2)$.	506	4.7
Edit	real	Normalized edit-distance.		100

of learned edges (the development and test set are described in Section 5). Graphs are always learned with the LP-Global algorithm.

Our main goal is to check whether the added features improve performance, and therefore we run the experiment both with and without the new features. In addition, we would like to test whether using different classifiers affects performance. Therefore, we run the experiments with a linear-kernel SVM, a square-kernel SVM, a Gaussian-kernel SVM, logistic regression, and naive Bayes. We use the SVMPerf package (Joachims 2005) to train the SVM classifiers and the Weka package (Hall et al. 2009) for logistic regression and naive Bayes.

6.2 Experiment Results

Table 13 describes the macro-average recall, precision, and F_1 of all classifiers both with and without the new features on the development set and test set. Using all features is denoted by X_{all}, and using the original features is denoted by X_{old}.

Table 13
Macro-average recall, precision, and F_1 on the development set and test set using the parameters that maximize F_1 of the learned edges over the development set.

	Development set			Test set		
Algorithm	Recall	Precision	F_1	Recall	Precision	F_1
Linear_{all}	48.1	31.9	**36.3**	51.7	37.7	40.3
Linear_{old}	40.3	33.3	34.8	47.2	42.2	41.1
Gaussian_{all}	41.8	32.4	35.1	48.0	41.1	40.7
Gaussian_{old}	41.1	31.2	33.9	50.3	39.7	40.5
Square_{all}	39.9	32.0	34.1	43.7	39.8	38.9
Square_{old}	38.0	31.6	32.9	50.2	41.0	**41.3**
Logistic_{all}	34.4	27.6	29.1	39.8	41.7	37.8
Logistic_{old}	39.3	31.2	33.5	45.4	40.9	39.9
Bayes_{all}	20.8	33.2	24.5	27.4	46.0	31.7
Bayes_{old}	20.3	34.9	24.6	26.4	45.4	30.9

Examining the results it does not appear that the new features improve performance. Whereas on the development set the new features add 1.2–1.5 F_1 points for all SVM classifiers, on the test set using the new features decreases performance for the linear and square classifiers. This shows that even if there is some slight increase in performance when using SVM on the development set, it is masked by the variance added in the process of parameter tuning. In general, including the new features does not yield substantial differences in performance.

Secondly, the SVM classifiers perform better than the logistic and naive Bayes classifiers. Using the more complex square and Gaussian kernels does not seem justified, however, as the differences between the various kernels are negligible. Therefore, in our analysis we will use a linear kernel SVM classifier.

Last, we note that although we use supervised learning rather than distant supervision, the results we get are slightly lower than those presented in Section 5. This is probably due to the fact that our manually annotated data set is rather small. Nevertheless, this shows that the quality of the distant supervision training set generated automatically from WordNet is reasonable.

Next, we perform analysis of the different features of the classifier to better understand the reasons for the negative result obtained.

6.3 Feature Analysis

We saw that the new features slightly improved performance for SVM classifiers on the development set, although no clear improvement was witnessed on the test set. To further check whether the new features carry useful information we measured the training set accuracy for each of the 12 training sets (leaving out each time one test-set graph). Using the new features improved the average training set accuracy from 71.6 to 72.3. More importantly, it improved performance consistently in all 12 training sets by 0.4–1.2 points. This strengthens our belief that the new features do carry a certain amount of information, but this information is too sparse to affect the overall

performance of the algorithm. In addition, notice that the absolute accuracy on the training set is low—72.3. This shows that separating entailment from non-entailment using the current set of features is challenging.

Next, we would like to perform analysis on each of the features. First, we perform an ablation test over the features by omitting each one of them and re-training the classifier Linear$_{all}$. In Table 14, the columns *ablation* F_1 and Δ show the F_1 obtained and the difference in performance from the Linear$_{all}$ classifier, which scored 40.3 F_1 points. Results show that there is no "bad" feature that deteriorates performance. For almost all features ablation causes a decrease in performance, although this decrease is relatively small. There are only four features for which ablation decreases performance by more than one point: three distributional similarity features, but also the new hypernym feature.

The next three columns in the table describe the precision and recall of the new boolean features. The column *Feature type* indicates whether we expect a feature to indicate entailment or non-entailment and the columns *Prec.* and *Recall* specify the

Table 14
Results of feature analysis. The second column denotes the proportion of manually annotated examples for which the feature value is non-zero. A detailed explanation of the other columns is provided in the body of the article.

Feature name	%	Ablation F_1	Δ	Feature type	Prec.	Recall	Classification F_1
h-b-B-pCt	8.2	39.3	−1				14.9
h-b-B-pC	6.9	39.5	−0.8				33.2
h-b-B-Ct	1.6	40.3	0				15.4
h-b-B-C	1.6	40.5	0.2				11.2
h-b-L-pCt	23.6	38.3	**−2.0**				**37.0**
h-b-L-pC	21.4	39.4	−0.9				35.2
h-b-L-Ct	9.7	40.1	−0.2				27.3
h-b-L-C	8.1	39.7	−0.6				14.1
h-u-B-Ct	1.0	39.4	−0.9				10.9
h-u-B-C	1.1	39.8	−0.5				12.6
h-u-L-Ct	6.1	39.8	−0.5				18.5
h-u-L-C	6.3	39.2	−1.1				19.3
R-b-L-l	22.5	40.1	−0.2				26.7
R-u-L-l	8.3	39.4	−0.9				23.2
R-u-B-l	1.9	39.8	−0.5				16.7
Lin & Pantel	8.8	38.7	−1.6				23.0
Hyper.	1.1	38.7	−1.6	+	37.1	4.9	9.7
Syno.	0.9	40.3	0	+	43.1	4.5	15.8
Co-hypo.	2.8	40.1	−0.2	−	82.0	2.5	17.9
WN ant.	0.06	39.8	−0.5	−	75.0	0.05	1.2
VO ant.	0.2	40.1	−0.2	−	96.0	0.2	2.2
WN ant. 2.	0.2	39.4	−0.9	−	59.1	0.1	2.7
VO ant. 2	0.7	40.2	−0.1	−	98.6	0.7	2.2
Derivation	0.7	39.5	−0.8	+	47.4	4.1	10.2
Entailment	0.2	39.7	−0.6	+	15.0	0.3	1.2
FRED	0.08	39.7	−0.6	+	77.8	0.8	3.2
NomLex	0.07	39.8	−0.5	+	75.0	0.7	3.3
VO strong.	1	39.4	−0.9	+	34.6	4	6.9
VO simil.	1.8	39.4	−0.9	+	28.8	6.1	12.5
Positive	2.7	39.8	−0.5	+	36.7	11.8	
BAP	4.7	40.1	−0.2				13.3
Edit	100	39.9	−0.4				15.5

precision and recall of that feature. For example, the feature *FRED* is a *positive* feature that we expect to support entailment, and indeed 77.8% of the gold-standard examples for which it is turned on are positive examples. It is turned on only in 0.8% of the positive examples, however. Similarly, *VO ant.* is a *negative* feature that we expect to support non-entailment, and indeed 96% of the gold-standard examples for which it is on are negative examples, but it is turned on in only 0.2% of the negative examples. The precision results are quite reasonable: For most positive features the precision is well over the proportion of positive examples in the gold standard, which is about 10% (except for the *Entailment* feature whose precision is only 15%). For the negative features it seems that the precision of VerbOcean features is very high (though they are sparse), and the precision of WordNet antonyms and co-hyponyms is lower. Looking at the recall we can see that the coverage of the boolean features is low.

The last column in the table describes results of training the classifier with a single feature. For each feature we train a linear kernel SVM, tune the sparsity parameter on the development set, and measure F_1 over the test set. Naturally, classifiers that are trained on sparse features yield low performance.

This column allows us once again (cf. Table 8) to examine the original distributional similarity features. There are three distributional similarity features that achieve F_1 of more than 30 points, and all three represent features using *pairs* of argument instantiations rather than treat each argument separately, as we have already witnessed in Section 5.

Note also that the feature *h-b-L-pCt*, which uses binary templates, the *Lin* similarity measure, and features that are pairs of CUI tuples, is the best feature both in terms of the ablation test and when it is used as a single feature for the classifier. The result obtained by this feature is only 3.3 points lower than that obtained when using the entire feature set. We believe this is for two reasons: First, the 16 distributional similarity features are correlated with one another and thus using all of them does not boost performance substantially. For example, the Pearson correlation coefficients between the features *h-b-B-pCt*, *h-b-B-Ct*, *h-b-L-pCt*, *h-b-L-Ct*, *h-u-B-Ct*, and *h-u-L-Ct* (all utilize CUI tuples) and *h-b-B-pC*, *h-b-B-C*, *h-b-L-pC*, *h-b-L-C*, *h-u-B-C*, and *h-u-L-C* (all use CUIs), respectively, are over 0.9. The second reason for gaining only 3.3 points by the remaining features is that, as discussed, the new set of features is relatively sparse.

To sum up, we suggest several hypotheses that explain our results and analysis:

- The new features are too sparse to substantially improve the performance of the local entailment classifier in our data set. This perhaps can be attributed to the nature of our domain-specific health-care corpus. In the future, we would like to examine the sparsity of these features in a general domain.
- Looking at the training set accuracy, ablations, and precision of the new features, it seems that the behavior of most of them is reasonable. Thus, it is possible that in a different learning scheme that does not use the resources as features the information they provide may become beneficial. For example, in a simple "back-off" approach one can use rules from precise resources to determine entailment, and apply a classifier only when no precise resource contains a relevant rule.
- In our corpus representing distributional similarity features with pairs of argument instantiations is better than treating each argument independently.

- Given the current training set accuracy and the sparsity of the new features, it is important to develop methods that gather large-scale information that is orthogonal to distributional similarity. In our opinion, the most promising direction for acquiring such rich information is by methods that look at co-occurrence of predicates or templates on the Web (Chklovski and Pantel 2004; Pekar 2008).

7. Conclusions and Future Work

This article presented a global optimization algorithm for learning entailment rules between predicates, represented as propositional templates. Most previous work on learning entailment rules between predicates focused on local learning methods, which consider each pair of predicates in isolation. To the best of our knowledge, this is the most comprehensive attempt to date to exploit global interactions between predicates for improving the set of learned entailment rules.

We modeled the problem as a graph learning problem, and searched for the best graph under a global transitivity constraint. Two objective functions were defined for the optimization procedure, one score-based and the other probabilistic, and we have shown that under certain conditions (specified in Appendix A) the score-based function can be interpreted probabilistically. This allowed us to use both margin as well as probabilistic classifiers for the underlying entailment classifier. We solved the optimization problem using Integer Linear Programming, which provides an optimal solution (compared to the greedy algorithm suggested by Snow, Jurafsky, and Ng [2006]), and demonstrated empirically that this method outperforms local algorithms as well as a state-of-the-art greedy optimization algorithm on the graph learning task. We also analyzed quantitatively and qualitatively the reasons for the improved performance of our global algorithm and performed detailed error analysis. Last, we experimented with various entailment classifiers that utilize different sets of features from many knowledge bases.

The experiments and analysis performed indicate that the current performance of the local entailment classifier needs to be improved. We believe that the most promising direction for improving the local classifier is to use methods that look for co-occurrence of predicates in sentences or documents on the Web, because these methods excel at identifying specific semantic relations. It is also possible to use other sources of information such as lexicographic resources, although this probably will require a learning scheme that is robust to the relatively low coverage of these resources. Increasing the size of the training corpus is also an important direction for improving the entailment classifier.

Another important direction for future work is to apply our algorithm to graphs that are larger by a few orders of magnitude than the focused entailment graphs dealt with in this article. This will introduce a challenge to our current optimization algorithm due to complexity issues, as our ILP contains $O(|V|^3)$ constraints. In addition, this will require careful handling of predicate ambiguity, which interferes with the transitivity of entailment and will become a pertinent issue in large graphs. Some first steps in this direction have already been carried out (Berant, Dagan, and Goldberger 2011).

In addition, our graphs currently contain a single type of edge, namely, the entailment relation. We would like to model more types of edges in the graph, representing additional semantic relations such as co-hyponymy, and to explicitly describe the interactions between the various types of edges, aiming to further improve the quality of the learned entailment rules.

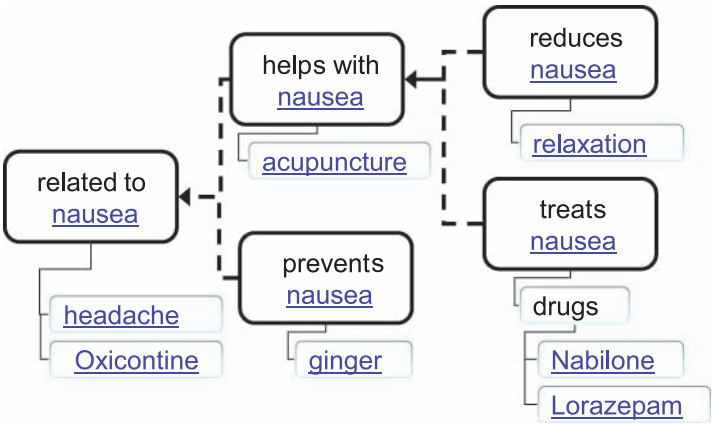

Figure 8
A hierarchical summary of propositions involving *nausea* as an argument, such as *headache is related to nausea*, *acupuncture helps with nausea*, and *Lorazepam treats nausea*.

Last, in Section 3.1 we mentioned that by merging strongly connected components in entailment graphs, hierarchies of predicates can be generated (recall Figure 1). As proposed by Berant, Dagan, and Goldberger (2010), we believe that these hierarchies can be useful not only in the context of semantic inference applications, but also in the field of faceted search and hierarchical text exploration (Stoica, Hearst, and Richardson 2007). Figure 8 exemplifies how a set of propositions can be presented to a user according to the hierarchy of predicates shown in Figure 1. In the field of faceted search, information is presented using a number of hierarchies, corresponding to different facets or dimensions of the data. One can easily use the hierarchy of predicates learned by our algorithm as an additional facet in the context of a text-exploration application. In future work, we intend to implement this application and perform user experiments to test whether adding this hierarchy facilitates exploration of textual information.

Appendix A: Derivation of the Probabilistic Objective Function

In this section we provide a full derivation for the probabilistic objective function given in Section 4.2.2. Given two nodes u and v from a set of nodes V, we denote by $I_{uv} = 1$ the event that u entails v, by F_{uv} the feature vector representing the ordered pair (u, v), and by F the set of feature vectors over all ordered pairs of nodes, that is, $F = \cup_{u \neq v} F_{uv}$. We wish to learn a set of edges E, such that the posterior probability $P(G|F)$ is maximized, where $G = (V, E)$. We assume that we have a "local" model estimating the edge posterior probability $P_{uv} = P(I_{uv} = 1|F_{uv})$. Because this model was trained over a balanced training set, the prior for the event that u entails v under the model is uniform: $P(I_{uv} = 1) = P(I_{uv} = 0) = \frac{1}{2}$. Using Bayes's rule we get:

$$P(I_{uv} = 1|F_{uv}) = \frac{P(I_{uv} = 1)}{P(F_{uv})} \cdot P(F_{uv}|I_{uv} = 1) = a \cdot P(F_{uv}|I_{uv} = 1) \quad \text{(A.1)}$$

$$P(I_{uv} = 0|F_{uv}) = \frac{P(I_{uv} = 0)}{P(F_{uv})} \cdot P(F_{uv}|I_{uv} = 0) = a \cdot P(F_{uv}|I_{uv} = 0) \quad \text{(A.2)}$$

where $a = \frac{1}{2 \cdot P(F_{uv})}$ is a constant with respect to any graph. Thus, we conclude that $P(I_{uv}|F_{uv}) = a \cdot P(F_{uv}|I_{uv})$. Next, we make three independence assumptions (the first two are following Snow, Jurafsky, and Ng [2006]):

$$P(F|G) = \prod_{u \neq v} P(F_{uv}|G) \tag{A.3}$$

$$P(F_{uv}|G) = P(F_{uv}|I_{uv}) \tag{A.4}$$

$$P(G) = \prod_{u \neq v} P(I_{uv}) \tag{A.5}$$

Assumption A.3 states that each feature vector is independent from other feature vectors given the graph. Assumption A.4 states that the features F_{uv} for the pair (u, v) are generated by a distribution depending only on whether entailment holds for (u, v). Last, Assumption A.5 states that edges are independent and the prior probability of a graph is a product of the prior probabilities of the edges. Using these assumptions and equations A.1 and A.2, we can now express the posterior $P(G|F)$:

$$P(G|F) \propto P(G) \cdot P(F|G) \tag{A.6}$$

$$= \prod_{u \neq v} [P(I_{uv}) \cdot P(F_{uv}|I_{uv})] \tag{A.7}$$

$$= \prod_{u \neq v} P(I_{uv}) \cdot \frac{P(I_{uv}|F_{uv})}{a} \tag{A.8}$$

$$\propto \prod_{u \neq v} P(I_{uv}) \cdot P_{uv} \tag{A.9}$$

$$= \prod_{(u,v) \in E} P(I_{uv} = 1) \cdot P_{uv} \cdot \prod_{(u,v) \notin E} P(I_{uv} = 0) \cdot (1 - P_{uv}) \tag{A.10}$$

Note that under the "local model" the prior for an edge in the graph was uniform, because the model was trained over a balanced training set. Generally, however, this is not the case, and thus we introduce an edge prior into the model when formulating the global objective function. Now, we can formulate P(G|F) as a linear function:

$$\hat{G} = \operatorname*{argmax}_G \prod_{(u,v) \in E} P(I_{uv} = 1) \cdot P_{uv} \cdot \prod_{(u,v) \notin E} P(I_{uv} = 0) \cdot (1 - P_{uv}) \tag{A.11}$$

$$= \operatorname*{argmax}_G \sum_{(u,v) \in E} \log(P_{uv} \cdot P(I_{uv} = 1)) + \sum_{(u,v) \notin E} \log[(1 - P_{uv}) \cdot P(I_{uv} = 0)] \tag{A.12}$$

$$= \operatorname*{argmax}_G \sum_{u \neq v} \left(I_{uv} \cdot \log(P_{uv} \cdot P(I_{uv} = 1)) + (1 - I_{uv}) \cdot \log[(1 - P_{uv}) \cdot P(I_{uv} = 0)] \right) \tag{A.13}$$

$$= \operatorname*{argmax}_G \sum_{u \neq v} \left(\log \frac{P_{uv} \cdot P(I_{uv} = 1)}{(1 - P_{uv}) \cdot P(I_{uv} = 0)} \cdot I_{uv} + (1 - P_{uv}) \cdot P(I_{uv} = 0) \right) \tag{A.14}$$

$$= \underset{G}{\operatorname{argmax}} \sum_{u \neq v} \log \frac{P_{uv}}{(1 - P_{uv})} \cdot I_{uv} + \log \eta \cdot |E| \tag{A.15}$$

In the last transition we omit $\sum_{u \neq v}(1 - P_{uv}) \cdot P(I_{uv} = 0)$, which is a constant with respect to the graph and denote the prior odds ratio by $\eta = \frac{P(I_{uv}=1)}{P(I_{uv}=0)}$. This leads to the final formulation described in Section 4.2.2.

Acknowledgments

We would like to thank Roy Bar-Haim, David Carmel, and the anonymous reviewers for their useful comments. We also thank Dafna Berant and the nine students who prepared the gold-standard data set. This work was developed under the collaboration of FBK-irst/University of Haifa and was partially supported by the Israel Science Foundation grant 1112/08. The first author is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship, and has carried out this research in partial fulfilment of the requirements for the Ph.D. degree.

References

Althaus, Ernst, Nikiforos Karamanis, and Alexander Koller. 2004. Computing locally coherent discourses. In *Proceedings of the ACL*, pages 399–406, Barcelona.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley framenet project. In *Proceedings of COLING-ACL*, pages 86–90, Montreal.

Bar-Haim, Roy, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*, pages 871–876, Vancouver.

Ben Aharon, Roni, Idan Szpektor, and Ido Dagan. 2010. Generating entailment rules from framenet. In *Proceedings of ACL*, pages 241–246, Uppsala.

Bentivogli, Luisa, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernarde Magnini. 2009. The fifth Pascal recognizing textual entailment challenge. In *Proceedings of TAC-09*, pages 14–24, Gaithersburg, MD.

Berant, Jonathan, Ido Dagan, and Jacob Goldberger. 2010. Global learning of focused entailment graphs. In *Proceedings of ACL*, pages 1220–1229, Uppsala.

Berant, Jonathan, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of ACL*, pages 610–619, Portland, OR.

Bhagat, Rahul, Patrick Pantel, and Eduard Hovy. 2007. LEDIR: An unsupervised algorithm for learning directionality of inference rules. In *Proceedings of EMNLP-CoNLL*, pages 161–170, Prague.

Budanitsky, Alexander and Graeme Hirst. 2006. Evaluating Wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Chklovski, Timothy and Patrick Pantel. 2004. VerbOcean: Mining the Web for fine-grained semantic verb relations. In *Proceedings of EMNLP*, pages 33–40, Barcelona.

Clark, Peter, William Murray, John Thompson, Phil Harrison, Jerry Hobbs, and Christiane Fellbaum. 2007. On the role of lexical and world knowledge in RTE3. In *Proceedings of the Workshop on Textual Entailment and Paraphrasing*, pages 54–59, Prague.

Clarke, James and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:273–381.

Cohen, William, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *Proceedings of IIWeb*, pages 73–78, Acapulco.

Connor, Michael and Dan Roth. 2007. Context sensitive paraphrasing with a single unsupervised classifier. In *Proceedings of ECML*, pages 104–115, Warsaw.

Coyne, Bob and Owen Rambow. 2009. Lexpar: A freely available English paraphrase lexicon automatically extracted from Framenet. In *Proceedings of the IEEE International Conference on Semantic Computing*, pages 53–58, Berkeley, CA.

Dagan, Ido, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):1–17.

Do, Quang and Dan Roth. 2010. Constraints based taxonomic relation classification. In *Proceedings of EMNLP*, pages 1099–1109, Cambridge, MA.

Fellbaum, Christiane. 1998a. A semantic network of English: The mother of all

wordNets. *Natural Language Engineering*, 32:209–220.
Fellbaum, Christiane, editor. 1998b. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, Cambridge, MA.
Finkel, Jenny R. and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of ACL-08: HLT, Short Papers*, pages 45–48, Columbus, OH.
Habash, Nizar and Bonnie Dorr. 2003. A categorial variation database for English. In *Proceedings of the NAACL*, pages 17–23, Edmonton.
Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18.
Harmeling, Stefan. 2009. Inferring textual entailment with a probabilistically sound calculus. *Natural Language Engineering*, 15(4):459–477.
Harris, Zellig. 1954. Distributional structure. *Word*, 10(23):146–162.
Joachims, Thorsten. 2005. A support vector method for multivariate performance measures. In *Proceedings of ICML*, pages 377–384, Bonn.
Kingsbury, Paul, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn TreeBank. In *Proceedings of HLT*, pages 252–256, San Diego, CA.
Kipper, Karin, Hoa T. Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of AAAI*, pages 691–696, Austin, TX.
Kotlerman, Lili, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16:359–389.
Lin, Dekang. 1998a. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*, pages 768–774, Montreal.
Lin, Dekang. 1998b. Dependency-based evaluation of Minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC*, pages 317–329, Granada.
Lin, Dekang and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.
Macleod, Catherine, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. Nomlex: A lexicon of nominalizations. In *Proceedings of Euralex*, pages 187–193, Liège.
Martins, Andre, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL*, pages 342–350, Singapore.
Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekeley, Veronika Zielinska, and Brian Young. 2004. The cross-breeding of dictionaries. In *Proceedings of LREC*, pages 1095–1098, Lisbon.
Mirkin, Shachar, Ido Dagan, and Maayan Gefet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *Proceedings of COLING-ACL*, pages 579–586, Sydney.
Nikulin, Vladimir. 2008. Classification of imbalanced data with random sets and mean-variance filtering. *International Journal of Data Warehousing and Mining*, 4(2):63–78.
Pekar, Viktor. 2008. Discovery of event entailment knowledge from text corpora. *Computer Speech & Language*, 22(1):1–16.
Raina, Rajat, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI*, pages 1099–1105, Pittsburgh, PA.
Riedel, Sebastian and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of EMNLP*, pages 129–137, Sydney.
Roth, Dan and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL*, pages 1–8, Boston, MA.
Schoenmackers, Stefan, Jesse Davis, Oren Etzioni, and Daniel S. Weld. 2010. Learning first-order horn clauses from Web text. In *Proceedings of EMNLP*, pages 1088–1098, Cambridge, MA.
Sekine, Satoshi. 2005. Automatic paraphrase discovery based on context and keywords between NE pairs. In *Proceedings of IWP*, pages 80–87, Jeju Island.
Siegel, Sidney and N. John Castellan. 1988. *Non-parametric Statistics for the Behavioral Sciences*. McGraw-Hill, New-York.
Smith, Noah and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*, pages 354–362, Ann Arbor, MI.

Snow, Rion, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of NIPS*, pages 1297–1304, Vancouver.

Snow, Rion, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of ACL*, pages 801–808, Prague.

Stoica, Emilia, Marti Hearst, and Megan Richardson. 2007. Automating creation of hierarchical faceted metadata structures. In *Proceedings of NAACL-HLT*, pages 244–251, Rochester, NY.

Szpektor, Idan and Ido Dagan. 2007. Learning canonical forms of entailment rules. In *Proceedings of RANLP*, pages 1–8, Borovetz.

Szpektor, Idan and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of COLING*, pages 849–856, Manchester.

Szpektor, Idan and Ido Dagan. 2009. Augmenting Wordnet-based inference with argument mapping. In *Proceedings of TextInfer*, pages 27–35, Singapore.

Szpektor, Idan, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling Web-based acquisition of entailment relations. In *Proceedings of EMNLP*, pages 41–48, Barcelona.

Van Hulse, Jason, Taghi Khoshgoftaar, and Amri Napolitano. 2007. Experimental perspectives on learning from imbalanced data. In *Proceedings of ICML*, pages 935–942, Corvallis, OR.

Vanderbei, Robert. 2008. *Linear Programming: Foundations and Extensions*. Springer, New-York.

Weeds, Julie and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of EMNLP*, pages 81–88, Sapporo.

Wilcoxon, Frank. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83.

Yannakakis, Mihalis. 1978. Node-and edge-deletion NP-complete problems. In *STOC '78: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 253–264, New York, NY.

Yates, Alexander and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.