Learning Typed Entailment Graphs with Global Soft Constraints

Citation for published version:

Digital Object Identifier (DOI):
10.1162/tacl_a_00250

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Transactions of the Association for Computational Linguistics

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Learning Typed Entailment Graphs with Global Soft Constraints

Mohammad Javad Hosseini⋆§ Nathanael Chambers** Siva Reddy† Xavier R. Holt‡ Shay B. Cohen⋆ Mark Johnson‡ and Mark Steedman⋆

⋆University of Edinburgh  ⎯ §The Alan Turing Institute, UK  **United States Naval Academy  ⎯ †Stanford University  ‡Macquarie University

javad.hosseini@ed.ac.uk, nchamber@usna.edu, sivar@stanford.edu {xavier.ricketts-holt,mark.johnson}@mq.edu.au {scohen,steedman}@inf.ed.ac.uk

Abstract

This paper presents a new method for learning typed entailment graphs from text. We extract predicate-argument structures from multiple-source news corpora, and compute local distributional similarity scores to learn entailments between predicates with typed arguments (e.g., person contracted disease). Previous work has used transitivity constraints to improve local decisions, but these constraints are intractable on large graphs. We instead propose a scalable method that learns globally consistent similarity scores based on new soft constraints that consider both the structures across typed entailment graphs and inside each graph. Learning takes only a few hours to run over 100K predicates and our results show large improvements on two entailment datasets. We further show improvements over paraphrases and entailments from the Paraphrase Database, and prior state-of-the-art entailment graphs. We show that the entailment graphs improve performance in a downstream task.

1 Introduction

Recognizing textual entailment and paraphrasing is critical to many core natural language processing applications such as question-answering and semantic parsing. The surface form of a sentence that answers a question such as “Does Verizon own Yahoo?” frequently does not directly correspond to the form of the question, but is rather a paraphrase or an expression such as “Verizon bought Yahoo”, that entails the answer. The lack of a well-established form-independent semantic representation for natural language is the most important single obstacle to bridging the gap between queries and text resources.

This paper seeks to learn meaning postulates (e.g., buying entails owning) that can be used to augment the standard form-dependent semantics. Our immediate goal is to learn entailment rules between typed predicates with two arguments, where the type of each predicate is determined by the types of its arguments. We construct typed entailment graphs, with typed predicates as nodes and entailment rules as edges. Figure 1 shows simple examples of such graphs with arguments of types company,company and person,location.

Entailment relations are detected by computing a similarity score between the typed predicates based on the distributional inclusion hypothesis, which states that a word (predicate) \( u \) entails another word (predicate) \( v \) if in any context that \( u \) can be used so can be \( v \) (Dagan et al., 1999; Gefen and Dagan, 2005; Herbelot and Ganesalingam, 2013; Kartsaklis and Sadrzadeh, 2016). Most previous work has taken a “local learning” approach (Lin, 1998; Weeds and Weir, 2003; Szpektor and Dagan, 2008; Schoenmackers et al., 2010), i.e., learning entailment rules independently from each other.

One problem facing local learning approaches is that many correct edges are not identified because of data sparsity and many wrong edges are spuriously identified as valid entailments. A “global learning” approach, where dependencies between entailment rules are taken into account, can improve the local decisions significantly. Berant et al. (2011) imposed transitivity constraints on the entailments, such that the inclusion of rules
$i \rightarrow j$ and $j \rightarrow k$ implies that of $i \rightarrow k$. While they showed transitivity constraints to be effective in learning entailment graphs, the Integer Linear Programming (ILP) solution of Berant et al. (2011) is not scalable beyond a few hundred nodes. In fact, the problem of finding a maximally weighted transitive subgraph of a graph with arbitrary edge weights is NP-hard (Berant et al., 2011).

This paper instead proposes a scalable solution that does not rely on transitivity closure, but instead uses two global soft constraints that maintain structural similarity both across and within each typed entailment graph (Figure 2). We introduce an unsupervised framework to learn globally consistent similarity scores given local similarity scores (§4). Our method is highly parallelizable and takes only a few hours to apply to more than 100K predicates.\footnote{We performed our experiments on a 32-core 2.3 GHz machine with 256GB of RAM.} \footnote{Our code, extracted binary relations and the learned entailment graphs are available at \url{https://github.com/mjhosseini/entGraph}.}

Our experiments (§6) show that the global scores improve significantly over local scores and outperform state-of-the-art entailment graphs on two standard entailment rule datasets (Berant et al., 2011; Holt, 2018). We ultimately intend the typed entailment graphs to provide a resource for entailment and paraphrase rules for use in semantic parsing and open domain question-answering, as has been done for similar resources such as the Paraphrase Database (PPDB; Ganitkevitch et al., 2013; Pavlick et al., 2015) in Wang et al. (2015); Dong et al. (2017).\footnote{Predicates inside each clique in the entailment graphs are considered to be paraphrases.} With that end in view, we have included a comparison with PPDB in our evaluation on the entailment datasets. We also show that the learned entailment rules improve performance on a question-answering task (§7) with no tuning or prior knowledge of the task.

2 Related Work

Our work is closely related to Berant et al. (2011), where entailment graphs are learned by imposing transitivity constraints on the entailment relations. However, the exact solution to the problem is not scalable beyond a few hundred predicates, while the number of predicates that we capture is two orders of magnitude larger (§5). Hence, it is necessary to resort to approximate methods based on assumptions concerning the graph structure. Berant et al. (2012) and Berant et al. (2015) propose Tree-Node-Fix (TNF), an approximation method that scales better by additionally assuming the entailment graphs are “Forest Reducible”, where a predicate cannot entail two (or more) predicates $j$ and $k$ such that neither $j \rightarrow k$ nor $k \rightarrow j$ (FRG assumption). However, the FRG assumption is not correct for many real-world domains. For example, a person visiting a place entails both arriving at that place and leaving that place, while the latter do not necessarily entail each other. Our work injects two other types of prior knowledge about the structure of the graph that are less expensive to incorporate and yield better results on entailment rule datasets.

Abend et al. (2014) learn entailment relations over multi-word predicates with different levels of compositionality. Pavlick et al. (2015) add variety of relations, including entailment, to phrase pairs in PPDB. This includes a broader range of entailment relations such as lexical entailment. In contrast to our method, these works rely on supervised data and take a local learning approach.

Another related strand of research is link prediction (Socher et al., 2013; Bordes et al., 2013; Riedel et al., 2013; Yang et al., 2015; Trouillon et al., 2016; Dettmers et al., 2018), where the source data are extractions from text, facts in knowledge bases, or both. Unlike our work, which directly learns entailment relations between predicates, these methods aim at predicting the source data, i.e., whether two entities have a particular relationship. The common wisdom is that entailment relations are by-product of these methods (Riedel et al., 2013). However, this assumption has not usually been explicitly evaluated. Explicit entailment rules provide explainable resources that can be used in downstream tasks. Our experiments show that our method significantly outperforms a state-of-the-art link prediction method.

3 Computing Local Similarity Scores

We first extract binary relations as predicate-argument pairs using a Combinatory Categorial Grammar (CCG; Steedman, 2000) semantic parser (§3.1). We map the arguments to their Wikipedia URLs using a named entity linker (§3.2). We extract types such as person and disease for each argument (§3.2). We then compute local similarity
scores between predicate pairs (§3.3).

3.1 Relation Extraction

The semantic parser of Reddy et al. (2014), GraphParser, is run on the NewsSpike corpus (Zhang and Weld, 2013) to extract binary relations between a predicate and its arguments from sentences. GraphParser uses CCG syntactic derivations and \( \lambda \)-calculus to convert sentences to neo-Davisonian semantics, a first-order logic that uses event identifiers (Parsons, 1990). For example, for the sentence, Obama visited Hawaii in 2012, GraphParser produces the logical form \( \exists e. \text{visit}_1(e, \text{Obama}) \land \text{visit}_2(e, \text{Hawaii}) \land \text{visit}_3(e, 2012) \), where \( e \) denotes an event. We will consider a relation for each pair of arguments, hence, there will be three relations for the above sentence: \( \text{visit}_{1,2} \) with arguments (Obama,Hawaii), \( \text{visit}_{1,in} \) with arguments (Obama,2012) and \( \text{visit}_{2,in} \) with arguments (Hawaii,2012). We currently only use extracted relations that involve two named entities or one named entity and a noun. We constrain the relations to have at least one named entity to reduce ambiguity in finding entailments.

We perform a few automatic post-processing steps on the output of the parser. First, we normalize the predicates by lemmatization of their head words. Passive predicates are mapped to active ones and we extract negations and particle verb predicates. Next, we discard unary relations and relations involving coordination of arguments. Finally, whenever we see a relation between a subject and an object, and a relation between object and a third argument connected by a prepositional phrase, we add a new relation between the subject and the third argument by concatenating the relation name with the object. For example, for the sentence China has a border with India, we extract a relation have border, with between China and India. We perform a similar process for PPs attached to VPs. Most of the light verbs and multi-word predicates will be extracted by the above post-processing (e.g., take care, of) which will recover many salient ternary relations.

While entailments and paraphrasing can benefit from n-ary relations, e.g., person visits a location in a time, we currently follow previous work (Lewis and Steedman, 2013a; Berant et al., 2015) in confining our attention to binary relations, leaving the construction of n-ary graphs to future work.

3.2 Linking and Typing Arguments

Entailment and paraphrasing depend on context. While using exact context is impractical in forming entailment graphs, many authors have used the type of the arguments to disambiguate polysemous predicates (Berant et al., 2011, 2015; Lewis and Steedman, 2013a; Lewis, 2014). Typing also reduces the size of the entailment graphs.

Since named entities can be referred to in many different ways, we use a named entity linking tool to normalize the named entities. In the experiments below, we use AIDALight (Nguyen et al., 2014), a fast and accurate named entity linker, to link named entities to their Wikipedia URLs (if any). We thus type all entities that can be grounded in Wikipedia. We first map the Wikipedia URL of the entities to Freebase (Bollacker et al., 2008). We select the most notable type of the entity from Freebase and map it to FIGER types (Ling and Weld, 2012) such as building, disease, person and location, using only the first level of the FIGER type hierarchy. For example, instead of event/sports_event, we use event as type. If an entity cannot be grounded in Wikipedia or its Freebase type does not have a mapping to FIGER, we assign the default type thing to it.

3.3 Local Distributional Similarities

For each typed predicate (e.g., \( \text{visit}_{1,2} \) with types person,location), we extract a feature vector. We use as feature types the set of argument pair strings (e.g., Obama-Hawaii) that instantiate the binary relations of the predicates. The value of each feature is the pointwise mutual information (PMI) between the predicate and the feature. We use the feature vectors to compute three local similarity scores (both symmetric and directional) between typed predicates: Weeds (Weeds and Weir, 2003), Lin (Lin, 1998), and Balanced Inclusion (BInc; Szpektor and Dagan, 2008) similarities.

4 Learning Globally Consistent Entailment Graphs

We learn globally consistent similarity scores based on local similarity scores. The global scores will be used to form typed entailment graphs.

\(^{49}\) out of 113 FIGER types.
4.1 Problem Formulation

Let $T$ be a set of types and $P$ be a set of predicates. We denote by $V(t_1, t_2)$ the set of typed predicates $p(t_1, t_2)$, where $t_1, t_2 \in T$ and $p \in P$. Each $p(t_1, t_2) \in V(t_1, t_2)$ takes as input arguments of types $t_1$ and $t_2$. An example of a typed predicate is $\text{win}_{1,2}(\text{team}, \text{event})$ that can be instantiated with $\text{win}_{1,2}(\text{Seahawks}, \text{team}, \text{Super Bowl}, \text{event})$.

We define $V(t_1, t_2) = V(t_1, t_2) \cup V(t_2, t_1)$. We often denote elements of $V(t_1, t_2)$ by $i$, $j$ and $k$, where each element is a typed predicate as above. For an $i=p(t_1, t_2) \in V(t_1, t_2)$, we denote by $\pi(i)=p$, $\tau_1(i)=t_1$ and $\tau_2(i)=t_2$. We compute distributional similarities between predicates with the same argument types. We denote by $W^0(t_1, t_2) \in [0,1]^{V(t_1, t_2) \times V(t_1, t_2)}$ the (sparse) matrix containing all local similarity scores $w_{ij}$ between predicates $i$ and $j$ with types $t_1$ and $t_2$, where $|V(t_1, t_2)|$ is the size of $V(t_1, t_2)$.

Predicates can entail each other with the same argument order (direct) or in the reverse order, i.e., $p(t_3, t_2)$ might entail $q(t_1, t_2)$ or $q(t_2, t_1)$. For the graphs with the same types (e.g., $t_1=t_2=\text{person}$), we keep two copies of the predicates one for each of the possible orderings. This allows us to model entailments with reverse argument orders, e.g., $\text{is son of}_{1,2}(\text{person1}.; \text{person2}) \rightarrow \text{is parent of}_{1,2}(\text{person1}.; \text{person2})$.

We define $V = \bigcup_{t_1, t_2} V(t_1, t_2)$, the set of all typed predicates, and $W^0$ as a block-diagonal matrix consisting of all the local similarity matrices $W^0(t_1, t_2)$. Similarly, we define $W(t_1, t_2)$ and $\bar{W}$ as the matrices consisting of globally consistent similarity scores $w_{ij}$ we wish to learn. The global similarity scores are used to form entailment graphs by thresholding $W$. For a $\delta > 0$, we define typed entailment graphs as $G_\delta(t_1, t_2) = (V(t_1, t_2), E_\delta(t_1, t_2))$, where $V(t_1, t_2)$ are the nodes and $E(t_1, t_2) = \{(i,j)|i,j \in V(t_1, t_2), w_{ij} \geq \delta\}$ are the edges of the entailment graphs.

4.2 Learning Algorithm

Existing approaches to learn entailment graphs from text miss many correct edges because of data sparsity, i.e., the lack of explicit evidence in the corpus that a predicate $i$ entails another predicate $j$. The goal of our method is to use evidence from the existing edges that have been assigned high confidence to predict missing ones, and remove spurious edges. We propose two global soft constraints that maintain structural similarity both across and within each typed entailment graph. The constraints are based on the following two observations.

First, it is standard to learn a separate typed entailment graph for each (plausible) type-pair because arguments provide necessary disambiguation for predicate meaning (Berant et al., 2011; Lewis and Steedman, 2013a,b; Berant et al., 2015). However, many entailment relations for which we have direct evidence only in a few subgraphs may in fact apply over many others (Figure 2A). For example, we may not have found direct evidence that mentions of a living_thing (e.g., a virus) triggering a disease are accompanied by mentions of the living_thing causing that disease (because of data sparsity), whereas we have found that mentions of a government_agency triggering an event are reliably accompanied by mentions of causing that event. While we show that typing is necessary to learning entailments (§6), we propose to learn all typed entailment graphs jointly.

Second, we encourage paraphrase predicates (where $i \rightarrow j$ and $j \rightarrow i$) to have the same patterns of entailment (Figure 2B), i.e. to entail and be entailed by the same predicates, global soft constraints that we call paraphrase resolution. Using these soft constraint, a missing entailment (e.g., medicine treats disease $\rightarrow$ medicine is useful for disease) can be identified by considering the en-

Figure 2: Learning entailments that are consistent (A) across different but related typed entailment graphs and (B) within each graph. $0 \leq \beta \leq 1$ determines how much different graphs are related. The dotted edges are missing, but will be recovered by considering relationships shown by across-graph (red) and within-graph (light blue) connections.
we expect the outgoing edges of typed entailment graphs to be similar. We name this global constraint, \( J(\mathbf{w} \geq 0, \beta \geq 0) = L_{\text{withinGraph}} + L_{\text{crossGraph}} + L_{\text{pResolution}} + \lambda_1 \| \mathbf{w} \|_1 \) (1)

\[ L_{\text{withinGraph}} = \sum_{i,j \in V} (w_{ij} - w_{ij})^2 \] (2)

\[ L_{\text{crossGraph}} = \frac{1}{2} \sum_{i,j \in V} \sum_{(i',j') \in N(i,j)} \beta \left( \pi(i), (\tau_1(i), \tau_2(i)), (\tau_1(i'), \tau_2(i')) \right) (w_{ij} - w_{i'j'})^2 + \frac{\lambda_2}{2} \| I - \beta \|_2^2 \] (3)

\[ L_{\text{pResolution}} = \frac{1}{2} \sum_{i,j \in T} \sum_{k \neq i, k \neq j} I_e(w_{ij}) I_e(w_{ji}) \left[ (w_{ik} - w_{kj})^2 + (w_{ki} - w_{kj})^2 \right] \] (4)

Figure 3: The objective function to jointly learn global scores \( \mathbf{w} \) and the compatibility function \( \beta \), given local scores \( \mathbf{w}^0 \). \( L_{\text{withinGraph}} \) encourages global and local scores to be close; \( L_{\text{crossGraph}} \) encourages similarities to be consistent between different typed entailment graphs; \( L_{\text{pResolution}} \) encourages paraphrase predicates to have the same pattern of entailment. We use an \( \ell_1 \) regularization penalty to remove entailments with low confidence.

Sharing entailments across different typed entailment graphs is only semantically correct for some predicates and types. In order to learn how we can generalize an entailment from one graph to another, we define a compatibility function \( \beta : P \times (T \times T) \times (T \times T) \rightarrow [0, 1] \). The function is defined for a predicate and two type pairs (Figure 2A). It specifies the extent of compatibility for a single predicate between different typed entailment graphs, with 1 being completely compatible and 0 being irrelevant. In particular \( \beta(p(t_1, t_2), (t_1', t_2')) \) determines how much we expect the outgoing edges of \( p(t_1, t_2) \) and \( p(t_1', t_2') \) to be similar. We constrain \( \beta \) to be symmetric between \( t_1, t_2 \) and \( t_1', t_2' \) as compatibility of outgoing edges of \( p(t_1, t_2) \) with \( p(t_1', t_2') \) should be the same as \( p(t_1', t_2) \) with \( p(t_1, t_2) \). We denote by \( \vec{\beta} \), a vectorization consisting of the values of \( \beta \) for all possible input predicates and types.

Note that the global similarity scores \( \mathbf{w} \) and the compatibility function \( \beta \) are not known in advance. Given local similarity scores \( \mathbf{w}^0 \), we learn \( \mathbf{w} \) and \( \vec{\beta} \) jointly. We minimize the loss function defined in Eq. 1 which consists of three soft constraints defined below and an \( \ell_1 \) regularization term (Figure 3).

\[ L_{\text{withinGraph}} \text{ Eq. 2} \] encourages global scores \( w_{ij} \) to be close to local scores \( w_{ij}^0 \), so that the global scores will not stray too far from the original scores.

\[ L_{\text{crossGraph}} \text{ Eq. 3} \] encourages each predicate’s entailments to be similar across typed entailment graphs (Figure 2A) if the predicates have similar neighbors. We penalize the difference of entailments in two different graphs, when the compatibility function is high. For each pair of typed predicates \((i, j) \in V(t_1, t_2)\), we define a set of neighbors (predicates with different types):

\[
N(i, j) = \left\{ (i', j') \in V(t_1', t_2') | t_1', t_2' \in T, (i', j') \neq (i, j), \pi(i) = \pi(i'), \pi(j) = \pi(j'), a(i, j) = a(i', j') \right\},
\] (5)

where \( a(i, j) \) is true if the argument orders of \( i \) and \( j \) match, and false otherwise. For each \((i', j') \in N(i, j)\), we penalize the difference of entailments by adding the term \( \beta(i)(w_{ij} - w_{i'j'})^2 \). We add a prior term on \( \vec{\beta} \) as \( \lambda_2 \| \vec{\beta} \|_2^2 \), where \( \vec{I} \) is a vector of the same size as \( \vec{\beta} \) with all 1s. Without the prior term (i.e., \( \lambda_2 = 0 \)), all the elements of \( \vec{\beta} \) will become zero. Increasing \( \lambda_2 \) will keep (some of the) elements of \( \vec{\beta} \) non-zero and encourages communications between related graphs.

\[ L_{\text{pResolution}} \text{ Eq. 4} \] denotes the paraphrase resolution global soft constraints that encourage paraphrase predicates to have the same patterns of entailments (Figure 2B). The function \( I_e(x) \) equals \( x \) if \( x > \varepsilon \) and zero, otherwise. Unlike \( L_{\text{crossGraph}} \) in Eq. 3, Eq. 4 operates on the edges within each graph. If both \( w_{ij} \) and \( w_{ji} \) are high, their incoming and outgoing edges from/to nodes \( k \) are encouraged to be similar. We name this global constraint,
paraphrase resolution, since it might add missing links (e.g., \(i \rightarrow k\)) if \(i\) and \(j\) are paraphrases of each other and \(j \rightarrow k\), or break the paraphrase relation, if the incoming and outgoing edges are very different.

We impose an \(\ell_1\) penalty on the elements of \(\mathbf{W}\) as \(\lambda_1 \| \mathbf{W} \|_1\), where \(\lambda_1\) is a nonnegative tuning hyperparameter that controls the strength of the penalty applied to the elements of \(\mathbf{W}\). This term removes entailments with low confidence from the entailment graphs. Note that Eq. 1 has \(\mathbf{W}\) and \(\mathbf{W}\) with low confidence from the entailment graphs. This term removes entailments with low confidence from the entailment graphs. Note that Eq. 1 has \(\mathbf{W}\) and average of \(\mathbf{W}\) across different typed entailment graphs (§5.4) as its special cases. The former is achieved by setting \(\lambda_1 = \lambda_2 = 0\) and \(\epsilon = 1\) and the latter by \(\lambda_1 = 0\), \(\lambda_2 = \infty\) and \(\epsilon = 1\). We do not explicitly weight the different components of the loss function, as the effect of \(L_{\text{crossGraph}}\) and \(L_{\text{pResolution}}\) can be controlled by \(\lambda_2\) and \(\epsilon\), respectively.

Eq. 1 can be interpreted as an inference problem in a Markov Random Field (MRF) (Kiefer and Snell, 1980), where the nodes of the MRF are the global scores \(w_{ij}\) and the parameters \(\vec{\beta}(p_i, t_{ij}, t_i', t_j')\) . The MRF will have five log-linear factor types: one unary factor type for \(L_{\text{withinGraph}}\), one three-variable factor type for the first term of \(L_{\text{crossGraph}}\) and a unary factor type for the prior on \(\vec{\beta}\), one four-variable factor type for \(L_{\text{pResolution}}\) and a unary factor type for the \(\ell_1\) regularization term. Figure 2 shows an example factor graph (unary factors are not shown for simplicity).

We learn \(\mathbf{W}\) and \(\vec{\beta}\) jointly using a message passing approach based on the Block Coordinate Descent method (Xu and Yin, 2013) . We initialize \(\mathbf{W} = \mathbf{W}^0\). Assuming that we know the global similarity scores \(\mathbf{W}\), we learn how much the entailments are compatible between different types (\(\vec{\beta}\)) and vice versa. Given \(\mathbf{W}\) fixed, each \(w_{ij}\) sends messages to the corresponding \(\vec{\beta}(\cdot)\) elements, which will be used to update \(\vec{\beta}\). Given \(\vec{\beta}\) fixed, we do one iteration of learning for each \(w_{ij}\). Each \(\vec{\beta}(\cdot)\) and \(w_{ij}\) elements send messages to the related elements in \(\mathbf{W}\), which will be in turn updated. Based on the update rules (Appendix A), we always have \(w_{ij} \leq 1\) and \(\vec{\beta} \leq \vec{1}\).

Each iteration of the learning method takes \(\mathcal{O}(\|\mathbf{W}\|_0|T|^2 + \sum_{i \in \mathcal{V}} (\|w_{i1}\|_0 + \|w_{i2}\|_0)^2)\) time, where \(\|\mathbf{W}\|_0\) is the number of nonzero elements of \(\mathbf{W}\) (number of edges in the current graph), \(|T|\) is the number of types and \(|w_{i1}|_0, |w_{i2}|_0\) is the number of nonzero elements of the \(i\)th row (column) of the matrix (out-degree and in-degree of the node \(i\)).\(^7\) In practice, learning converges after 5 iterations of full updates. The method is highly parallelizable, and our efficient implementation does the learning in only a few hours.

5 Experimental Setup

We extract binary relations from a multiple-source news corpus (§5.1) and compute local and global scores. We form entailment graphs based on the similarity scores and test our model on two entailment rules datasets (§5.2). We then discuss parameter tuning (§5.3) and baseline systems (§5.4).

5.1 Training Corpus: Multiple-Source News

We use the multiple-source NewsSpike corpus of Zhang and Weld (2013). NewsSpike was deliberately built to include different articles from different sources describing identical news stories. They scraped RSS news feeds from January-February 2013 and linked them to full stories collected through a web search of the RSS titles. The corpus contains 550K news articles (20M sentences). Since this corpus contains multiple sources covering the same events, it is well-suited to our purpose of learning entailment and paraphrase relations.

We extracted 29M binary relations using the procedure in Section 3.1. In our experiments, we used two cutoffs within each typed subgraph to reduce the effect of noise in the corpus: (1) remove any argument-pair that is observed with less than \(C_1 = 3\) unique predicates; (2) remove any predicate that is observed with less than \(C_2 = 3\) unique argument-pairs. This leaves us with \(|P| = 101K\) unique predicates in 346 entailment graphs. The maximum graph size is 53K nodes \(^8\) and the total number of non-zero local scores in all graphs is 66M. In the future, we plan to test our method on an even larger corpus, but preliminary experiments suggest that data sparsity will persist regardless of the corpus size, due to the power law distribution of the terms. We compared our extractions qualitatively with Stanford Open IE (Etzioni et al., 2011; Angeli et al., 2015). Our CCG-based extraction generated noticeably better results in our experiments, the total number of edges is \(\approx 0.01|V|^2\) and most of predicate pairs are seen in less than 20 subgraphs, instead of \(|T|^2\).

\(^7\)There are 4 graphs with more than 20K nodes, 3 graphs with 10K to 20K nodes, and 16 graphs with 1K to 10K nodes.
tions for longer sentences with long-range dependencies such as those involving coordination.

5.2 Evaluation Entailment Datasets

Levy/Holt’s Entailment Dataset  Levy and Dagan (2016) proposed a new annotation method (and a new dataset) for collecting relational inference data in context. Their method removes a major bias in other inference datasets such as Zeichner’s (Zeichner et al., 2012), where candidate entailments were selected using a directional similarity measure. Levy & Dagan form questions of the type which city (qtype), is located near (qrel), mountains (qarg)? and hold only in one direction. Holt (2018) designed a task to crowd-annotate the dataset by a) adding the reverse entailment and provide possible answers of the form A is surrounded by (a), mountains (aarg). Annotators are shown a question with multiple possible answers, where a is masked by qtype to reduce the bias towards world knowledge. If the annotator indicates the answer as True (False), it is interpreted that the predicate in the answer entails (does not entail) the predicate in the question.

While the Levy entailment dataset removes bias, a recent evaluation identified high labeling error rate for entailments that hold only in one direction (Holt, 2018). Holt analyzed 150 positive examples and showed that 33% of the claimed entailments are correct only in the opposite direction, while 15% do not entail in any direction. Holt (2018) designed a task to crowd-annotate the dataset by a) adding the reverse entailment (q→a) for each original positive entailment (a→q) in Levy’s dataset; and b) directly asking the annotators if a positive example (or its reverse) is an entailment or not (as opposed to relying on a factoid question). We test our method on this re-annotated dataset of 18,407 examples (3,916 positive and 14,491 negative), which we refer to as Levy/Holt. We run our CCG based binary relation extraction on the examples and perform our typing procedure (§3.2) on a (e.g., Kyoto) and aarg (e.g., mountains) to find the types of the arguments. We split the re-annotated dataset into dev (30%) and test (70%) such that all the examples with the same qtype and qrel are assigned to only one of the sets.

Berant’s Entailment Dataset  Berant et al. (2011) annotated all the edges of 10 typed entailment graphs based on the predicates in their corpus. The dataset contains 3,427 edges (positive), and 35,585 non-edges (negative). We evaluate our method on all the examples of Berant’s entailment dataset. The types of this dataset do not match with FIGER types, but we perform a simple hand-mapping between their types and FIGER types.10

5.3 Parameter Tuning

We selected λ1=.01 and ε=.3 based on preliminary experiments on the dev set of Levy/Holt’s dataset. The hyperparameter λ2 is selected from \{0, 0.01, 0.1, 0.5, 1, 1.5, 2, 10, 100\}.11 We do not tune λ2 for Berant’s dataset. We instead use the selected value based on the Levy/Holt dev set. In all our experiments, we remove any local score w_i = .01. We show precision-recall curves by changing the threshold δ on the similarity scores.

5.4 Comparison

We test our model by ablation of the global soft constraints \(L_{crossGraph}\) and \(L_{pResolution}\), testing simple baselines to resolve sparsity and comparing to the state-of-the-art resources. We also compare with two distributional approaches that can be used to predict predicate similarity. We compare the following models and resources.

CG is our novel model with both global soft constraints \(L_{crossGraph}\) and \(L_{pResolution}\). CG is our novel model without \(L_{pResolution}\). Local is the local distributional similarities without any change. AVG is the average of local scores across all the entailment graphs that contain both predicates in an entailment of interest. We set \(λ_2 = ∞\) which forces all the values of \(β\) to be 1, hence resulting in a uniform average of local scores. Untyped scores are local scores learned without types. We set the cutoffs \(C_1=20\) and \(C_2=20\) to have a graph with total number of edges similar to the typed entailment graphs.

ConvE scores are cosine similarities of low-dimensional predicate representations learned by ConvE (Dettmers et al., 2018), a state-of-the-art model for link prediction. ConvE is a multi-layer convolutional network model that is highly parameter efficient. We learn 200-dimensional vectors for each predicate (and argument) by applying ConvE to the set of extractions of the above untyped graph. We learned embeddings for each predicate and its reverse to handle examples where the argument order of the two predicates are different.

---

1010 mappings in total (e.g., animal to living_thing).
11The selected value was usually around 1.5.
ent. Additionally, we tried TransE (Bordes et al., 2013), another link prediction method which despite of its simplicity, produces very competitive results in knowledge base completion. However, we do not present its full results as they were worse than ConvE.\footnote{We also tried the average of GloVe embeddings (Pennington et al., 2014) of the words in each predicate, but the results were worse than ConvE.}

**PPDB** is based on the Paraphrase Database (PPDB) of Pavlick et al. (2015). We accept an example as entailment if it is labeled as a paraphrase or entailment in the PPDB XL lexical or phrasal collections.\footnote{We also tested the largest collection (XXXL), but the precision was very low on Berant’s dataset (below 30%).} Berant\_ILP is based on the entailment graphs of Berant et al. (2011).\footnote{We also tested (Berant et al., 2015), but do not report the results as they are very similar.} For Berant’s dataset, we directly compared our results to the ones reported in Berant et al. (2011). For Levy/Holt’s dataset, we used publicly available entailment rules derived from Berant et al. (2011) that gives us one point of precision and recall in the plots. While the rules are typed and can be applied in a context sensitive manner, ignoring the types and applying the rules out of context yields much better results (Levy and Dagan, 2016). This is attributable to both the non-standard types used by Berant et al. (2011) and also the general data sparsity issue.

In all our experiments, we first test a set of rule-based constraints introduced by Berant et al. (2011) on the examples before the prediction by our methods. In the experiments on Levy/Holt’s dataset, in order to maintain compatibility with Levy and Dagan (2016), we also run the lemma based heuristic process used by them before applying our methods. We do not apply the lemma based process on Berant’s dataset in order to compare with Berant et al’s (2011) reported results directly. In experiments with CG\_PR and CG, if the typed entailment graph corresponding to an example does not have one or both predicates, we resort to the average score between all typed entailment graphs.

### 6 Results and Discussion

To test the efficacy of our globally consistent entailment graphs, we compare them with the baseline systems in Section 6.1. We test the effect of approximating transitivity constraints in Section 6.2. Section 6.3 concerns error analysis.

#### 6.1 Globally Consistent Entailment Graphs

We test our method using three distributional similarity measures: Weeds similarity (Weeds and Weir, 2003), Lin similarity (Lin, 1998) and Balanced Inclusion (BInc; Szpektor and Dagan, 2008). The first two similarity measures are symmetric,\footnote{Weeds similarity is the harmonic average of Weeds precision and Weeds recall, hence a symmetric measure.} while BInc is directional. Figures 4A and 4B show precision-recall curves of the different methods on Levy/Holt’s and Berant’s datasets, respectively, using BInc. We show the full curve for BInc as it is directional and on the development portion of Levy/Holt’s dataset, it yields better results than Weeds and Lin.

In addition, Table 1 shows the area under the precision-recall curve (AUC) for all variants of the three similarity measures. Note that each method covers a different range of precisions and recalls. We compute AUC for precisions in the range $[0.5, 1]$, because predictions with precision better than random guess are more important for end applications such as question-answering and semantic parsing. For each similarity measure, we tested statistical significance between the methods using bootstrap resampling with $10K$ experiments (Efron and Tibshirani, 1985; Koehn, 2004). In Table 1, the best result for each dataset and similarity measure is boldfaced. If the difference of another model with the best result is not significantly different with $p$-value $<.05$, the second model is also boldfaced.

<table>
<thead>
<tr>
<th></th>
<th>local</th>
<th>untyped</th>
<th>AVG</th>
<th>CG</th>
<th>CG_PR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LEVY/HOLT’S</strong> dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BInc</td>
<td>.076</td>
<td>.127</td>
<td>.157</td>
<td>.162</td>
<td>.165</td>
</tr>
<tr>
<td>Lin</td>
<td>.074</td>
<td>.120</td>
<td>.146</td>
<td>.151</td>
<td>.149</td>
</tr>
<tr>
<td>Weed</td>
<td>.073</td>
<td>.115</td>
<td>.143</td>
<td>.149</td>
<td>.147</td>
</tr>
<tr>
<td>ConvE</td>
<td>-</td>
<td>.112</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>local</th>
<th>untyped</th>
<th>AVG</th>
<th>CG</th>
<th>CG_PR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BERANT’S</strong> dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BInc</td>
<td>.138</td>
<td>.167</td>
<td>.144</td>
<td>.177</td>
<td>.179</td>
</tr>
<tr>
<td>Lin</td>
<td>.147</td>
<td>.158</td>
<td>.172</td>
<td>.186</td>
<td>.189</td>
</tr>
<tr>
<td>Weed</td>
<td>.146</td>
<td>.154</td>
<td>.171</td>
<td>.184</td>
<td>.187</td>
</tr>
<tr>
<td>ConvE</td>
<td>-</td>
<td>.144</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Area under precision-recall curve (for precision $>0.5$) for different variants of similarity measures: local, untyped, AVG, crossGraph (CG) and crossGraph + pResolution (CG\_PR). We report results on two datasets. Bold indicates stat significance (see text).
Among the distributional similarities based on BInc, BInc_CG_PR outperforms all the other models in both datasets. In comparison to BInc score’s AUC, we observe more than 100% improvement on Levy/Holt’s dataset and about 30% improvement on Berant’s. Given the consistent gains, our proposed model appears to alleviate the data sparsity and the noise inherent to local scores. Our method also outperforms PPDB and Berant_ILP on both datasets. The second best performing model is BInc_CG, which improves the results significantly, especially on Berant’s dataset, over the BInc_AVG (AUC of .177 vs .144). This confirms that learning what subset of entailments should be generalized across different typed entailment graphs ($\beta$) is effective.

The untyped models yield a single large entailment graph. It contains (noisy) edges that are not found in smaller typed entailment graphs. Despite the noise, untyped models for all three similarity measures still perform better than the typed ones in terms of AUC. However, they do worse in the high-precision range. For example, BInc_untyped is worse than BInc for precision > 0.85. The AVG models do surprisingly well (only about 0.5 to 3.5 below CG_PR in terms of AUC), but note that only a subset of the typed entailment graphs might have (untyped) predicates $p$ and $q$ of interest (usually not more than 10 typed entailment graphs out of 367 graphs). Therefore, the AVG models are generally expected to outperform the untyped ones (with only one exception in our experiments), as typing has refined the entailments and averaging just improves the recall. Comparison of CG_PR with CG models confirms that explicitly encouraging paraphrase predicates to have the same patterns of entailment is effective. It improves the results for BInc score, which is a directional similarity measure. We also tested applying the paraphrase resolution soft constraints alone, but the
differences with the local scores were not statistically significant. This suggests that the paraphrase resolution is more helpful when similarities are transferred between graphs, as this can cause inconsistencies around the predicates with transferred similarities, which are then resolved by the paraphrase resolution constraints.

The results of the distributional representations learned by ConvE are worse than most other methods. We attribute this outcome to the fact that a) while entailment relations are directional, these methods are symmetric; b) the learned embeddings are optimized for tasks other than entailment or paraphrase detection; and c) the embeddings are learned regardless of argument types. However, even the BInc_untyped baseline outperforms ConvE, showing that it is important to use a directional measure that directly models entailment. We hypothesize that learning predicate representations based on the distributional inclusion hypotheses which do not have the above limitations might yield better results.

6.2 Effect of Transitivity Constraints

Our largest graph has 53K nodes, we thus tested approximate methods instead of the ILP to close entailment relations under transitivity (§2). The approximate TNF method of Berant et al. (2011) did not scale to the size of our graphs with moderate sparsity parameters. Berant et al. (2015) also present a heuristic method, High-To-Low Forest Reducible Graph (HTL-FRG), which gets slightly better results than TNF on their dataset, and which scales to graphs of the size we work with.16 We applied the HTL-FRG method to the globally consistent similarity scores (BInc.CG.PR_HTL) and changed the threshold on the scores to get a precision-recall curve. Figures 4C and 4D show the results of this method on Levy/Holt’s and Berant’s datasets. Our experiments show, in contrast to the results of Berant et al. (2015), that the HTL-FRG method leads to worse results when applied to our global scores. This result is caused both by the use of heuristic methods in place of globally optimizing via ILP, and by the removal of many valid edges arising from the fact that the FRG assumption is not correct for many real-world domains.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive</td>
<td></td>
</tr>
<tr>
<td>Spurious correlation (57%)</td>
<td>Microsoft released Internet Explorer → Internet Explorer was developed by Microsoft</td>
</tr>
<tr>
<td>Relation normalization (31%)</td>
<td>The pain may be relieved by aspirin → The pain can be treated with aspirin</td>
</tr>
<tr>
<td>Lemma based process &amp; parsing (12%)</td>
<td>President Kennedy came to Texas → President Kennedy came from Texas</td>
</tr>
<tr>
<td>False Negative</td>
<td></td>
</tr>
<tr>
<td>Sparsity (93%)</td>
<td>Cape town lies at the foot of mountains → Cape town is located near mountains</td>
</tr>
<tr>
<td>Wrong label &amp; parsing (7%)</td>
<td>Horses are imported from Australia → Horses are native to Australia</td>
</tr>
</tbody>
</table>

Table 2: Examples of different error categories and relative frequencies. The cause of errors is boldfaced.

6.3 Error Analysis

We analyzed 100 false positive (FP) and 100 false negative (FN) randomly selected examples (using BInc.CG_ST results on Levy/Holt’s dataset and at the precision level of Berant_ILP, i.e. 0.76). We present our findings in Table 2. Most of the FN errors are due to data sparsity, but a few errors are due to wrong labeling of the data and parsing errors. More than half of the FP errors are because of spurious correlations in the data that are captured by the similarity scores, but are not judged to constitute entailment by the human judges. About one third of the FP errors are because of the normalization we currently perform on the relations, e.g., we remove modals and auxiliaries. The remaining errors are mostly due to parsing and our use of Levy and Dagan’s (2016) lemma based heuristic process.

7 Extrinsic Evaluation

To further test the utility of explicit entailment rules, we evaluate the learned rules on an extrinsic task: answer selection for machine reading comprehension on NewsQA, a dataset that contains questions about CNN articles (Trischler et al., 2017). Machine reading comprehension is usually evaluated by posing questions about a text passage and then assessing the answers of a system (Trischler et al., 2017). The datasets that are used for this task are often in the form of (document, question, answer) triples, where an-
The board hailed Romney for his solid credentials. Researchers announced this week that they’ve found a new gene, ALS6, which is responsible for …

One out of every 17 children under 3 years old in America has a food allergy, and some will outgrow their sensitivities. The reported compromise could itself run afoul of European labor law, opening the way for foreign workers … … Barnes & Noble CEO William Lynch said as he unveiled his company’s Nook Tablet on Monday. The report said opium has accounted for more than half of Afghanistan’s gross domestic product in 2007.

Table 3: Examples where explicit entailment relations improve the rankings. The related words are boldfaced.

<table>
<thead>
<tr>
<th>ACC</th>
<th>MRR</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISF</td>
<td>36.18</td>
<td>48.99</td>
</tr>
<tr>
<td>ISFEnt</td>
<td>37.61</td>
<td>50.06</td>
</tr>
</tbody>
</table>

Table 4: Results (in percentage) for answer selection on the NewsQA dataset.

The accuracy results of Narayan et al. (2018) are not consistent with their own MRR and MAP (ACC>MRR in come cases), as they break ties between ISF scores differ-
\[
\begin{align*}
    w_{ij} &= 1(c_{ij} > \lambda_1)(c_{ij} - \lambda_1)/\tau_{ij} \\
    c_{ij} &= w_{ij}^0 + \sum_{(i',j') \in N(i,j)} \beta(\cdot)w_{i'j'} - \left(\mathbb{1}(w_{ij} > \varepsilon)I_\varepsilon(w_{ji}) + \sum_{k \in V(\tau_1(i),\tau_2(i))} [(w_{ik} - w_{jk})^2 + (w_{ki} - w_{kj})^2]\right) \\
    &+ 2 \sum_{k \in V(\tau_1(i),\tau_2(i))} I_\varepsilon(w_{jik})I_\varepsilon(w_{kj})w_{ik} + I_\varepsilon(w_{jik})I_\varepsilon(w_{ki})w_{kj} \\
    \tau_{ij} &= 1 + \sum_{(i',j') \in N(i,j)} \beta(\cdot) + 2 \sum_{k \in V(\tau_1(i),\tau_2(i))} I_\varepsilon(w_{jik})I_\varepsilon(w_{kj}) + I_\varepsilon(w_{jik})I_\varepsilon(w_{ki}) \\
    \beta(\cdot) &= I_0\left(1 - \left(\sum_{j \in V(\tau_1(i),\tau_2(i))}(w_{ij} - w_{i'j'})^2\right)/\lambda_2\right).
\end{align*}
\]

Figure 5: The update rules for \(w_{ij}\) and \(\beta(\cdot)\).

8 Conclusions and Future Work

We have introduced a scalable framework to learn typed entailment graphs directly from text. We use global soft constraints to learn globally consistent entailment scores for entailment relations. Our experiments show that generalizing in this way across different but related typed entailment graphs significantly improves performance over local similarity scores on two standard text entailment datasets. We show around 100\% increase in AUC on Levy/Holt’s dataset and 30\% on Berant’s dataset. The method also outperforms PPDB and the prior state-of-the-art entailment graph-building approach due to Berant et al. (2011). Paraphrase Resolution further improves the results. We have in addition showed the utility of entailment rules on answer selection for machine reading comprehension.

In the future, we plan to show that the global soft constraints developed in this paper can be extended to other structural properties of entailment graphs such as transitivity. Future work might also look at entailment relation learning and link prediction tasks jointly. The entailment graphs can be used to improve relation extraction, similar to Eichler et al. (2017), but covering more relations. In addition, we intend to collapse cliques in the entailment graphs to phrase clusters with a single relation identifier, and to replace the form-dependent lexical semantics of the CCG parser with these form-independent relations (Lewis and Steedman, 2013a) and to use the entailment graphs to derive meaning postulates for use in tasks such as question-answering and construction of knowledge-graphs from text (Lewis and Steedman, 2014).

Appendix A

Figure 5 shows the update rules of the learning algorithm. The global similarity scores \(w_{ij}\) are updated using Eq. 6, where \(c_{ij}\) and \(\tau_{ij}\) are defined in Eq. 7 and Eq. 8, respectively. \(\mathbb{1}(x)\) equals 1 if the condition \(x\) is satisfied and zero, otherwise. The compatibility functions \(\beta(\cdot)\) are updated using Eq. 9.

Acknowledgements

We thank Thomas Kober and Li Dong for helpful comments and feedback on the work, Reg-
gie Long for preliminary experiments on openIE extractions, and Ronald Cardenas for providing baseline code for the NewsQA experiments. The authors would also like to thank Katrin Erk and the three anonymous reviewers for their valuable feedback. This work was supported in part by the Alan Turing Institute under the EPSRC grant EP/N510129/1. The experiments were made possible by Microsoft’s donation of Azure credits to The Alan Turing Institute. The research was supported in part by ERC Advanced Fellowship GA 742137 SEMANTAX, a Google faculty award, a Bloomberg L.P. Gift award, and a University of Edinburgh/Huawei Technologies award to Steedman. Chambers was supported in part by the National Science Foundation under Grant IIS-1617952. Steedman and Johnson were supported by the Australian Research Council’s Discovery Projects funding scheme (project number DP160102156).

References


