In Computing Meaning, Harry Bunt, Johan Bos, and Stephen Pulman (Eds.), Vol. 4, pp. 27--48, Berlin 2013. Springer.

A Formal Approach to Linking Logical Form and Vector-Space Lexical Semantics

Dan Garrette, Katrin Erk, and Raymond Mooney

Abstract First-order logic provides a powerful and flexible mechanism for representing natural language semantics. However, it is an open question of how best to integrate it with uncertain, weighted knowledge, for example regarding word meaning. This paper describes a mapping between predicates of logical form and points in a vector space. This mapping is then used to project distributional inferences to inference rules in logical form. We then describe first steps of an approach that uses this mapping to recast first-order semantics into the probabilistic models that are part of Statistical Relational AI. Specifically, we show how Discourse Representation Structures can be combined with distributional models for word meaning inside a Markov Logic Network and used to successfully perform inferences that take advantage of logical concepts such as negation and factivity as well as weighted information on word meaning in context.

1 Introduction

Logic-based representations of natural language meaning have a long history (Montague, 1970; Kamp and Reyle, 1993). Representing the meaning of language in a first-order logical form is appealing because it provides a powerful and flexible way to express even complex propositions. However, systems built solely using first-order logical forms tend to be very brittle as they have no way of integrating uncertain knowledge. They, therefore, tend to have high precision at the cost of low recall (Bos and Markert, 2005).

Recent advances in computational linguistics have yielded robust methods that use statistically-driven weighted models. For example, distributional models of word

Dan Garrette
University of Texas at Austin, e-mail: dhg@cs.utexas.edu

Katrin Erk
University of Texas at Austin e-mail: katrin.erk@mail.utexas.edu

Raymond Mooney
University of Texas at Austin e-mail: mooney@cs.utexas.edu

meaning have been used successfully to judge paraphrase appropriateness by representing the meaning of a word in context as a point in a high-dimensional semantics space (Erk and Padó, 2008; Thater et al., 2010; Reisinger and Mooney, 2010; Dinu and Lapata, 2010; Van de Cruys et al., 2011). However, these models only address word meaning, and do not address the question of providing meaning representations for complete sentences. It is a long-standing open question how best to integrate the weighted or probabilistic information coming from such modules with logic-based representations in a way that allows for reasoning over both. See, for example, Hobbs et al. (1993).

The goal of this work is to establish a formal system for combining logic-based meaning representations with weighted information into a single unified framework. This will allow us to obtain the best of both situations: we will have the full expressivity of first-order logic and be able to reason with probabilities. We believe that this will allow for a more complete and robust approach to natural language understanding.

While this is a large and complex task, this paper proposes first steps toward our goal by presenting a mechanism for injecting distributional word-similarity information from a vector space into a first-order logical form. We define a mapping from predicate symbols of logical form to points in vector space. Our main aim in linking logical form to a vector space in this paper is to project inferences from the vector space to logical form. The inference rules that we use are based on substitutability. In a suitably constructed distributional representation, distributional similarity between two words or expressions A and B indicates that B can be substituted for A in text (Lin and Pantel, 2001). This can be described through an inference rule $A \rightarrow B$. Distributional information can also be used to determine the degree η to which the rule applies in a given sentence context (Szpektor et al., 2008; Mitchell and Lapata, 2008; Erk and Padó, 2008; Thater et al., 2010; Reisinger and Mooney, 2010; Dinu and Lapata, 2010; Van de Cruys et al., 2011). This degree η can be used as a weight on the inference rule $A \rightarrow B$.

In this paper, we first present our formal framework for projecting inferences from vector space to logical form. We then show how that framework can be applied to a real logical language and vector space to address issues of ambiguity in word meaning. Finally, we show how the weighted inference rules produced by our approach interact appropriately with the first-order logical form to produce correct inferences.

Our implementation uses Markov Logic Networks (MLN) (Richardson and Domingos, 2006) as the underlying engine for probabilistic inference. We are able to demonstrate that an MLN is able to properly integrate the first-order logical representation and weighted inference rules so that inferences involving correct word sense are assessed as being highly probable, inferences involving incorrect word sense are determined to be low probability, and inferences that violate hard logical rules are determined to have the lowest probability.

2 Background

Textual entailment. Recognizing Textual Entailment (RTE) is the task of determining whether one natural language text, the *premise*, implies another, the *hypothesis*. For evaluation of our system, we have chosen to use a variation on RTE in which we assess the relative probability of entailment for a of set of hypotheses.

We have chosen textual entailment as the mode of evaluation for our approach because it offers a good framework for testing whether a system performs correct analyses and thus draws the right inferences from a given text. As an example, consider (1) below.

(1) *p:* The spill left a stain.
h1: The spill resulted in a stain.
h2:* The spill fled a stain.
h3:* The spill did not result in a stain.

Here, hypothesis *h1* is a valid entailment, and should be judged to have high probability by the system. Hypothesis *h2* should have lower probability since it uses the wrong sense of *leave* and *h3* should be low probability because the logical operator *not* has reversed the meaning of the premise statement.

While the most prominent forum using textual entailment is the Recognizing Textual Entailment (RTE) challenge (Dagan et al., 2005), the RTE datasets do not test the phenomena in which we are interested. For example, in order to evaluate our system's ability to determine word meaning in context, the RTE pair would have to specifically test word sense confusion by having a word's context in the hypothesis be different from the context of the premise. However, this simply does not occur in the RTE corpora. In order to properly test our phenomena, we construct hand-tailored premises and hypotheses based on real-world texts.

Logic-based semantics. Boxer (Bos et al., 2004) is a software package for wide-coverage semantic analysis that provides semantic representations in the form of Discourse Representation Structures (Kamp and Reyle, 1993). It builds on the C&C CCG parser (Clark and Curran, 2004).

Bos and Markert (2005) describe a system for Recognizing Textual Entailment (RTE) that uses Boxer to convert both the premise and hypothesis of an RTE pair into first-order logical semantic representations and then uses a theorem prover to check for logical entailment.

Distributional models for lexical meaning. Distributional models describe the meaning of a word through the context in which it appears (Landauer and Dumais, 1997; Lund and Burgess, 1996), where contexts can be documents, other words, or snippets of syntactic structure. Based on the hypothesis that words that are similar in meaning will occur in similar contexts (Harris, 1954; Firth, 1957), distributional models predict semantic similarity between words based on distributional similarity. They can be learned in an unsupervised fashion. Recently distributional models have been used to predict the applicability of paraphrases in context (Erk and Padó, 2008; Thater et al., 2010; Reisinger and Mooney, 2010; Dinu and Lapata, 2010; Van de Cruys et al., 2011). For example, in “The spill left a stain”, *result in* is a better paraphrase for *leave* than *flee*, because of the context of *spill* and *stain*. In the sentence “The suspect left the country”, the opposite is true: *flee* is a better paraphrase. Usually, the distributional representation for a word mixes all its usages (senses). For the paraphrase appropriateness task, these representations are then reweighted, extended, or filtered to focus on contextually appropriate usages.

Markov Logic.

In order to perform logical inference with weights, we draw from the large and active body of work related to Statistical Relational AI (Getoor and Taskar, 2007). Specifically, we make use of Markov Logic Networks (MLNs) (Richardson and Domingos, 2006) which employ weighted graphical models to represent first-order logical formulas. MLNs are appropriate for our approach because they provide an elegant method of assigning weights to first-order logical rules, combining a diverse set of inference rules, and performing probabilistic inference.

An MLN consists of a set of weighted first-order clauses. It provides a way of softening first-order logic by making situations in which not all clauses are satisfied less likely, but not impossible (Richardson and Domingos, 2006). More formally, if X is the set of all propositions describing a world (i.e. the set of all ground atoms), $\mathcal{F}$ is the set of all clauses in the MLN, w_i is the weight associated with clause $f_i \in \mathcal{F}$, $\mathcal{G}_{f_i}$ is the set of all possible groundings of clause f_i, and $\mathcal{Z}$ is the normalization constant, then the probability of a particular truth assignment $\mathbf{x}$ to the variables in X is defined as:

$$P(X = \mathbf{x}) = \frac{1}{\mathcal{Z}} \exp\left(\sum_{f_i \in \mathcal{F}} w_i \sum_{g \in \mathcal{G}_{f_i}} g(\mathbf{x})\right) = \frac{1}{\mathcal{Z}} \exp\left(\sum_{f_i \in \mathcal{F}} w_i n_i(\mathbf{x})\right)$$

where $g(\mathbf{x})$ is 1 if g is satisfied and 0 otherwise, and $n_i(\mathbf{x}) = \sum_{g \in \mathcal{G}_{f_i}} g(\mathbf{x})$ is the number of groundings of f_i that are satisfied given the current truth assignment to the variables in X. This means that the probability of a truth assignment rises exponentially with the number of groundings that are satisfied.

Markov Logic has been used previously in other NLP applications (e.g. Poon and Domingos (2009)). However, this paper differs in that it is an attempt to represent deep logical semantics in an MLN.

While it is possible to learn rule weights in an MLN directly from training data, our approach at this time focuses on incorporating weights computed by external knowledge sources. Weights for word meaning rules are computed from the distributional model of lexical meaning and then injected into the MLN. Rules governing implicativity are given infinite weight (hard constraints).

We use the open source software package Alchemy (Kok et al., 2005) to perform MLN inference.

3 Linking logical form and vector spaces

In this section we define a link between logical form and vector space representations through a mapping function that connects predicates in logical form to points in vector space. Gärdenfors (2004) uses the interpretation function for this purpose, such that logical formulas are interpreted over vector space representations. However, the *conceptual spaces* that he uses are not distributional. Their dimensions are qualities, like the hue and saturation of a color or the taste of a fruit. Points in a conceptual space are, therefore, potential entities. In contrast, the vector spaces that we use are distributional in nature, and, therefore, cannot be interpreted as potential entities. A point in such a space is a potential word, defined through its observed contexts. For this reason, we define the link between logical form and vector space through a second mapping function independent of the interpretation function, which we call the *lexical mapping* function.

Lexical mapping and inference projection

Let V be a vector space whose dimensions stand for elements of textual context. We also write V for the set of points in the space. We assume that each word is represented as a point in vector space.[1] The central relation in vector spaces is semantic similarity. We represent this through a *similarity function*

$$\text{sim} : V \times V \to [0,1]$$

that maps each pair of points in vector space to their degree of similarity. While most similarity functions in the literature are symmetric, such that $\text{sim}(\vec{v},\vec{w}) = \text{sim}(\vec{w},\vec{v})$, our definition also accommodates asymmetric similarity measures like Kotlerman et al. (2010).

We link logical form and a vector space through a function that maps every predicate symbol to a point in space. Let $\mathcal{L}$ be a logical language. For each $n \geq 0$, let the set of n-ary predicate symbols of $\mathcal{L}$ be $\mathcal{P}^n_{\mathcal{L}}$, and let $\mathcal{P}_{\mathcal{L}} = \cup_{n \geq 0} \mathcal{P}^n_{\mathcal{L}}$. Let V be a vector space. Then a *lexical mapping function* from $\mathcal{L}$ to V is a function $\ell : \mathcal{P}_{\mathcal{L}} \to V$.

[1] The assumption of a single vector per word is made for the sake of simplicity. If we want to cover models in which each word is represented through multiple vectors (Reisinger and Mooney, 2010; Dinu and Lapata, 2010), this can be done through straightforward extensions of the definitions given here.

A central property of distributional vector spaces is that they can predict similarity in meaning based on similarity in observed contexts (Harris, 1954). Lin and Pantel (2001) point out that in suitably constrained distributional representations, distributional similarity indicates substitutability in text. If two words v and w are similar in their observed contexts, then w can be substituted for v in texts. This can be written as an inference rule $v \rightarrow w$, weighted by $\mathrm{sim}(\vec{v}, \vec{w})$.

We use this same idea to project inference rules from vector space to logical form through the lexical mapping function. If the lexical mapping function maps the n-ary predicate P to $\vec{v}$ and the n-ary predicate Q to $\vec{w}$, and $\mathrm{sim}(\vec{v}, \vec{w}) = \eta$, then we obtain the weighted inference rule $\forall x_1, \ldots, x_n[P(x_1, \ldots, x_n) \rightarrow Q(x_1, \ldots x_n)]$ with weight η. More generally, let $\mathcal{L}$ be a logical language with lexical mapping ℓ to a vector space V. Let sim be the similarity function on V. For all $Q \in \mathcal{P}_{\mathcal{L}}$ and $\mathcal{Q} \subseteq \mathcal{P}_{\mathcal{L}}$, let $\zeta(Q, \mathcal{Q}) \subseteq \mathcal{Q}$. Then the *inference projection* for the predicate $P \in \mathcal{P}^n_{\mathcal{L}}$ is

$$
\begin{aligned}
\Pi_{\mathrm{sim},\zeta,\ell}(P) = \{(F, \eta) \mid \exists Q \in \zeta(P, \mathcal{P}^n_{\mathcal{L}})\, [\, & \\
& F = \forall x_1, \ldots, x_n[P(x_1, \ldots, x_n) \rightarrow Q(x_1, \ldots, x_n)], \\
& \eta = \mathrm{sim}\big(\ell(P), \ell(Q)\big)\ ]\}
\end{aligned}
$$

That is, the inference projection for P is the set of all weighted inference rules (F, η) predicted by the vector space that let us infer some other predicate Q from P. Additionally, we may have information on the inferences that we are willing to project that is not encoded in the vector space. For example we may only want to consider predicates Q that stand for paraphrases of P. For this reason, the function ζ can be used to limit the predicates Q considered for the right-hand sides of rules. If $\zeta(P, \mathcal{P}^n_{\mathcal{L}}) = \mathcal{P}^n_{\mathcal{L}}$, then a rule will be generated for every $Q \in \mathcal{P}^n_{\mathcal{L}}$.

Addressing polysemy

When a word is polysemous, this affects the applicability of vector space-based inference rules. Consider the rule $\forall e[\mathit{fix}(e) \rightarrow \mathit{correct}(e)]$ (any fixing event is a correcting event): This rule applies in contexts like “fix a problem”, but not in contexts like “fix the date”. We therefore need to take context into account when considering inference rule applicability. We do this by computing vector representations for word meaning in context, and predicting rule applicability based on these context-specific vectors. We follow the literature on vector space representations for word meaning in context (Erk and Padó, 2008; Thater et al., 2010; Reisinger and Mooney, 2010; Dinu and Lapata, 2010; Van de Cruys et al., 2011) in assuming that a word’s context-specific meaning is a function of its out-of-context representation and the context. The context may consist of a single item or multiple items, and (syntactic or semantic) relations to the target word may also play a role (Erk and Padó, 2008; Thater et al., 2010; Van de Cruys et al., 2011).

We first define what we mean by a context. Given a vector space V and a finite set R of semantic relations, the set $C(V,R)$ *of contexts over* V *and* R consists of all finite sets of pairs from $V \times R$. That is, we describe the context in which a target word occurs as a finite set of pairs $(\vec{v}, r)$ of a context item $\vec{v}$ represented as a point in vector space, and the relation r between the context item and the target. For a word w in a context $c \in C(V,R)$, the context-specific meaning $\vec{w}_c$ of w is a function of the out-of-context vector $\vec{w}$ for w and the context c:

$$\vec{w}_c = \alpha(\vec{w}, c)$$

The function α is a *contextualization function* with signature $\alpha : V \times C(V,R) \rightarrow V$.

This definition of contextualization functions is similar to the framework of Mitchell and Lapata (2008), who define the meaning $\vec{p}$ of a two-word phrase $p = vw$ as a function of the vectors for v and w, and their syntactic relation r in the text: $\vec{p} = f(\vec{v}, \vec{w}, r, K)$, where f is some function, and K is background knowledge. However, we use contextualization functions to compute the meaning of a word in context, rather than the meaning of a phrase. We map predicate symbols to points in space, and predicate symbols need to map to word meanings, not phrase meanings. Also, Mitchell and Lapata only consider the case of two-word phrases, while we allow for arbitrary-size contexts.

In existing approaches to computing word meaning in context, bag-of-words representations or syntactic parses of the sentence context are used to compute the contextualization. In contrast, we use the logical form representation, through a function that maps a logic formula to a context in $C(V,R)$. Given a logical language $\mathcal{L}$, a vector space V, and set R of semantic relations, a *context mapping* is a function that computes the context $c \in C(V,R)$ of a predicate P in a formula G as

$$c = \kappa(P, G)$$

The signature of a context mapping function is $\kappa : \mathcal{P}_{\mathcal{L}} \times \mathcal{L} \rightarrow C(V,R)$.

We can now compute a context-specific vector space representation $\vec{w}_{P,G}$ for a predicate P in a formula G from the context-independent vector $\ell(P)$ and the context $\kappa(P,G)$. It is

$$\vec{w}_{P,G} = \alpha\big(\ell(P), \kappa(P,G)\big)$$

To obtain an inference projection for P that takes into account its context in the formula G, we adapt the lexical mapping function. Given a lexical mapping ℓ, let $\ell_{[Q/\vec{v}]}$ be the function that is exactly like ℓ except that it maps Q to $\vec{v}$. Let $\Pi_{\mathrm{sim},\zeta,\ell}$ be an inference projection for vector space V and logical language $\mathcal{L}$, let α be a contextualization function on V and R, and κ a context mapping from $\mathcal{L}$ to $C(V,R)$. Then the *contextualized inference projection* for predicate $P \in \mathcal{P}^n_{\mathcal{L}}$ in formula $G \in \mathcal{L}$ is

$$\Pi^G_{\mathrm{sim},\zeta,\ell}(P) = \Pi_{\mathrm{sim},\zeta,\ell_{[P/\alpha(\ell(P),\kappa(P,G))]}}(P)$$

In this contextualized inference projection, any rule $\forall x_1, \ldots, x_n[P(x_1, \ldots, x_n) \rightarrow Q(x_1, \ldots, x_n)]$ is weighted by similarity $\text{sim}(\alpha(\ell(P), \kappa(P,G)), \ell(Q))$ between the context-specific vector for P and the vector for Q. This follows common practice in vector space models of word meaning in context of computing a context-specific representation of the target, but not the paraphrase candidate. But if the paraphrase candidate is polysemous, it may be useful to compute a representation for it that is also specific to the sentence context at hand (Erk and Padó, 2010). We can do this by defining a lexical mapping $\gamma^{P,G}$ specific to predicate P and formula G by $\gamma^{P,G}(Q) = \alpha\big(\ell(Q), \kappa(P,G)\big)$. Then we can compute the contextualized inference projection of P as $\Pi^{G}_{\text{sim},\zeta,\ell}(P) = \Pi_{\text{sim},\zeta,\gamma^{P,G}}(P)$.

In computational semantics, polysemy is mostly addressed by using multiple predicates. For example, for the noun "bank" there would be predicates bank_1, bank_2 to cover the financial and riverside senses of the word. In contrast, we use a separate predicate for each word token, but these predicates are not associated with any particular fixed senses. Instead, we vary the lexical mapping of a predicate based on the formula that it appears in: A predicate P in a formula G is mapped to the vector $\alpha\big(\ell(P), \kappa(P,G)\big)$, which depends on G. We make this change for two reasons. First, a system that uses distinct predicates bank_1, bank_2 has to rely on an external word sense disambiguation system that decides, during semantics construction, which of the senses to use. In contrast, we determine lexical meaning based on the overall semantic representation of a sentence, directly linking sentence semantics and lexical semantics. Second, in the case of polysemy, the senses to distinguish are not always that clear. For example, for a noun like "onion", should the vegetable sense and the plant/bulb sense be separate (Krishnamurthy and Nicholls, 2000)? Through the vector space model, we can model word meaning in context without ever referring to distinct dictionary senses (Erk, 2010). But if we do not want to consider a fixed list of senses for a word w, then we also cannot represent its meanings through a fixed list of predicates.

4 Transforming natural language text to logical form

In transforming natural language text to logical form, we build on the software package Boxer (Bos et al., 2004). Boxer is an extension to the C&C parser (Clark and Curran, 2004) that transforms a parsed discourse of one or more sentences into a semantic representation. Boxer outputs the meaning of each discourse as a Discourse Representation Structure (DRS) that closely resembles the structures described by Kamp and Reyle (1993).

We chose to use Boxer for two main reasons. First, Boxer is a wide-coverage system that can deal with arbitrary text. Second, the DRSs that Boxer produces are close to the standard first-order logical forms that are required for use by the MLN software package Alchemy. Our system interprets discourses with Boxer, augments the resulting logical forms by adding inference rules, and outputs a format that the MLN software Alchemy can read.

5 Ambiguity in word meaning

In order for our system to be able to make correct natural language inferences, it must be able to handle paraphrasing. For example, in order to license the entailment pair in (2), the system must recognize that "owns" is a valid paraphrase for "has", and that a "car" is type of "vehicle":

(2) *p:* Ed owns a car.
h: Ed has a vehicle.

We address this problem as described in Section 3: We use distributional information to generate inferences stating, for example, that "has" can be substituted for "owns". This inference is weighted by the degree to which "owns", in the context in which it is used in (2), is similar to "has". To integrate these inference rules with the logical form representations of sentences like (2), we use the formalism introduced in Section 3. We now describe how we instantiate it in the current paper.

First, we generate a vector space V. We have chosen to implement a very simple vector space based on a bag-of-words representation of context. To ensure that the entries in the vector space correspond to the predicates in our logical forms, we first lemmatize all sentences in our corpus using the same lemmatization process as Boxer. The features used by V are the N most frequent lemmas, excluding stopwords. To calculate the vector in V for a lemma, we count the number of times the lemma appears in the same sentence as each feature, and then calculate the pointwise mutual information (PMI) between the lemma and each feature. The resulting PMI values for each feature are used as the vector for the lemma.

As the *similarity function* sim on our vector space, we use cosine similarity. For two vectors $\vec{v}$ and $\vec{w}$, their similarity is

$$\mathrm{sim}(\vec{v},\vec{w}) = \mathit{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{\|\vec{v}\|\;\|\vec{w}\|}$$

Logical forms in our system are generated by Boxer, so our logical language $\mathcal{L}$ is the set of formulas that may be returned from Boxer (modulo some modifications described in Section 6). Likewise, the set of predicate symbols $\mathcal{P}_{\mathcal{L}}$ are the predicates generated by Boxer. Boxer's predicates, as represented by the `pred` relation in Boxer's Prolog output,[2] consist of a word lemma and a token index indicating the

[2] See `http://svn.ask.it.usyd.edu.au/trac/candc/wiki/DRSs` for the detailed grammar of Boxer DRS output.

original token that generated that predicate. Our *lexical mapping* function maps each predicate symbol to the vector that represents the lemma portion of the predicate.

In order to assess the similarity between a word's context and a possible replacement word, we must define a *context mapping* that generates a context from a predicate $P \in \mathcal{P}_{\mathcal{L}}$ and a formula $G \in \mathcal{L}$. For the current paper we use the simplest possible definition for κ, which ignores semantic relations. We define the context of P as the vectors of all predicates Q that occur in the same sentence as P. Since every predicate in a logical form returned by Boxer is indexed with the sentence from which it was generated, we can define a simple context mapping that defines a predicate's context solely in terms of the other predicates generated by Boxer for that sentence.

$$\begin{aligned}\kappa(P,G) = \{(\textit{same-sentence}, \ell(Q)) \mid\ & Q \text{ is a predicate found in } G,\\ & Q\text{'s sentence index} = P\text{'s sentence index, and}\\ & Q \neq P\}\end{aligned}$$

Note that the only predicates Q that are used are those derived from the lemmas of words found in the text. Meta-predicates representing relations such as *agent*, *patient*, and *theme* are not included.

The context mapping κ computes a context for a predicate P occurring in a formula G. Next we require a *contextualization function* that uses the context returned by κ to compute a context-specific vector for P. Again we use the simplest instantiation possible. Our contextualization function just computes the sum of the vectors for each lemma in the context

$$\alpha(\vec{v},c) = \sum_{(r_i,\vec{w}_i)\in c} \vec{w}_i$$

Other, more complex instantiations of κ and α are possible. We comment on this further in Section 8.

Based on these definitions, we compute the *contextualized inference projection* $\Pi^G_{\text{sim},\zeta,\ell}(P)$, the set of weighted inference rules mapping predicate P to its potential replacements, as described in Section 3.

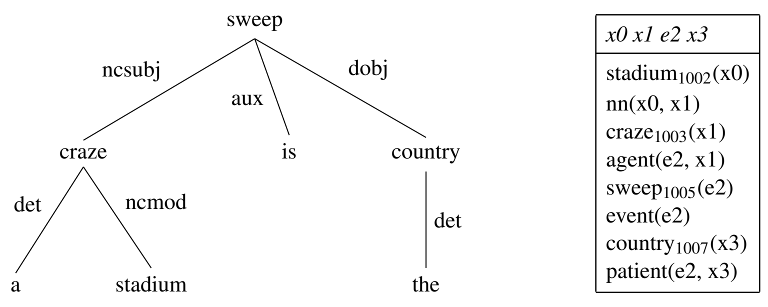
Finally, in order to limit the number of inference rules generated in the inference projection, we define a restriction function ζ that specifies, for a predicate $P \in \mathcal{P}^n_{\mathcal{L}}$, which of the predicates in $\mathcal{P}^n_{\mathcal{L}}$ may serve as replacements. Our system uses WordNet (Miller, 2009) to restrict substitutions only to those predicates representing synonyms or hypernyms of the lemma underlying P. So, for a predicate $P \in \mathcal{P}^n_{\mathcal{L}}$ and a set of predicates $\mathcal{Q} \subseteq \mathcal{P}^n_{\mathcal{L}}$, we define ζ as

$$\zeta(P,\mathcal{Q}) = \{Q \in \mathcal{Q} \mid Q\text{'s lemma is a synonym of, a hypernym of, or equal to } P\text{'s}\}$$

A lexical ambiguity example

Assume we have sentence (3), which is parsed by C&C and translated into DRT by Boxer, as shown in Figure 1.

(3) *p:* A stadium craze is sweeping the country.
h1: A craze is covering the nation.
h2:* A craze is brushing the nation.



(a) Dependency output from C&C (b) DRT output from Boxer

Fig. 1: Dependency parse tree and DRT interpretation of the premise in (3)

The DRS in Figure 1b, a formula of logical language $\mathcal{L}$, shall be denoted by G. Formula G contains a unary predicate $sweep_{1005}$. In order to generate weighted substitution rules for $sweep_{1005}$, we calculate the *contextualized inference projection* of $sweep_{1005}$: the set of inference rules mapping $sweep_{1005}$ to each (unary) predicate $Q \in \mathcal{P}^1_{\mathcal{L}}$, with each rule weighted by the similarity of the vector representing the context of $sweep_{1005}$ in G to the vector representing the replacement Q. This is

$$\begin{aligned}
&\Pi^G_{\text{sim},\zeta,\ell}(sweep_{1005}) = \\
&\qquad \{(F,\eta) \mid \exists Q \in \zeta(P,\mathcal{P}^1_{\mathcal{L}})\ [\\
&\qquad\qquad F = \forall x.[sweep_{1005}(x) \to Q(x)] \text{ and} \\
&\qquad\qquad \eta = \text{sim}\big(\alpha(\ell(sweep_{1005}), \kappa(sweep_{1005}, G)), \ell(Q)\big)\ ]\}
\end{aligned}$$

Let us assume that our logical language $\mathcal{L}$ also includes unary predicates $cover_{2004}$ and $brush_{3004}$ and that the lemmas *cover* and *brush* are known to be synonyms of *sweep* (though from different senses). In other words,

$$\{cover_{2004},\ brush_{3004}\} \in \zeta(sweep_{1005},\ \mathcal{P}^1_{\mathcal{L}})$$

So, in the calculation of $\Pi^{G}_{\text{sim},\zeta\ell}(sweep_{1005})$, we will generate weighted inference rules (F, η) for both $cover_{2004}$ and $brush_{3004}$. This will allow us to calculate the probability of inference for both hypotheses in (3).

We look first at $cover_{2004}$. The rule formula F is instantiated simply as

$$\forall x.[sweep_{1005}(x) \rightarrow cover_{2004}(x)]$$

The weight η is the similarity between the context of $sweep_{1005}$ in G, and $cover_{2004}$. The context vector for $sweep_{1005}$ is calculated as

$$\alpha(\ell(sweep_{1005}), \kappa(sweep_{1005}, G))$$

Since we defined the lexical mapping $\ell(P)$ to simply return the vector from V for the lemma portion of the predicate P, $\ell(sweep_{1005}) = \overrightarrow{sweep}$ and $\ell(cover_{2004}) = \overrightarrow{cover}$.

The context of P in G, $\kappa(P, G)$ is the set of a set of predicates and their relations to P, so

$$\begin{aligned}\kappa(sweep_{1005}, G) = \{&(\ell(stadium_{1002}),\ \textit{same-sentence})\} \\ &(\ell(craze_{1003}),\ \textit{same-sentence}), \\ &(\ell(country_{1007}),\ \textit{same-sentence}), \\ = \{&(\overrightarrow{stadium},\ \textit{same-sentence}), \\ &(\overrightarrow{craze},\ \textit{same-sentence}), \\ &(\overrightarrow{country},\ \textit{same-sentence})\}\end{aligned}$$

We defined our contextualization function $\alpha(\vec{v}, c)$ to be the vector sum of word vectors from the context c, so

$$\begin{aligned}\alpha(\ell(sweep_{1005}), \kappa(sweep_{1005}, G)) &= \alpha(\overrightarrow{sweep},\ \{(\overrightarrow{stadium},\ \textit{same-sentence}), \\ &\qquad (\overrightarrow{craze},\ \textit{same-sentence}), \\ &\qquad (\overrightarrow{country},\ \textit{same-sentence})\}) \\ &= \overrightarrow{stadium} + \overrightarrow{craze} + \overrightarrow{country}\end{aligned}$$

Finally, since we have the vector representing the context of $sweep_{1005}$ in G and the vector representing the replacement predicate $cover_{2004}$, we can compute the weight, η for our inference rule $\forall x.[sweep_{1005}(x) \rightarrow cover_{2004}(x)]$ as

$$\begin{aligned}&\text{sim}\big(\alpha(\ell(sweep_{1005}), \kappa(sweep_{1005}, G)), \ell(Q)\big) \\ &\qquad = \text{sim}\big(\overrightarrow{stadium} + \overrightarrow{craze} + \overrightarrow{country},\ \overrightarrow{cover}\big) \\ &\qquad = cosine\big(\overrightarrow{stadium} + \overrightarrow{craze} + \overrightarrow{country},\ \overrightarrow{cover}\big)\end{aligned}$$

Likewise, the rule for replacing $sweep_{1005}$ by $brush_{3004}$ would be $\forall x.[sweep_{1005}(x) \rightarrow brush_{3004}(x)]$ weighted by $cosine\big(\overrightarrow{stadium}+\overrightarrow{craze}+\overrightarrow{country},\ \overrightarrow{brush}\big)$.

Since, $cosine\big(\overrightarrow{stadium} + \overrightarrow{craze} + \overrightarrow{country},\ \overrightarrow{cover}\big) > cosine\big(\overrightarrow{stadium} + \overrightarrow{craze} + \overrightarrow{country},\ \overrightarrow{brush}\big)$, *cover* is considered to be a better replacement for *sweep* than *brush* in the sentence "A stadium craze is sweeping the country". Thus, the rule $\forall x.[sweep_{1005}(x) \rightarrow cover_{2004}(x)]$ will be given more consideration during inference, and hypothesis *h1* will be determined to be more probable than *h2*.

Hypernymy

According to our definition of ζ above, we construct inference rules of the form $\forall x_1,\ldots,x_n[P(x_1,\ldots,x_n) \rightarrow Q(x_1,\ldots x_n)]$ where Q is a synonym or hypernym of P. Thus, for two synonyms A and B, we will generate rules $A \rightarrow B$ and $B \rightarrow A$. However, for hypernym relationships, we only construct the inference rule entailing *up* the hierarchy: from the hyponym to the hypernym. This is important for licensing correct inferences. Consider example (4).

(4) *p:* Ed owns a car.
h: Ed has a vehicle.

Here the inference is valid since a *car* is a type of *vehicle*. For this pair, our system will generate the rule $\forall x[car(x) \rightarrow vehicle(x)]$ and assign a weight based on the similarity of the lemma *vehicle* to the context of *car* in the premise sentence. However, an inference in the reverse direction of (4) would be invalid, which is why we do not generate the reverse inference rule.

With hypernymy, we can see how our system naturally integrates logical phenomena with distributional information. In example (4), the distributional similarity between *vehicle* and the context of *car* affects the overall probability of inference for the pair. However, it does not override the logical requirements imposed by the hypernym relationship: if the premise and hypothesis were reversed then it would not matter how similar the words were since the inference would be impossible.

The logical rules generated for hypernyms work properly with other logical aspects as well. For example, in (5) below we can see that the direction of entailment along the hypernym hierarchy is reversed when the words appear in negative contexts. Our system handles this correctly.

(5) *p:* Ed does not own a vehicle.
h: Ed does not have a car.

6 Implicativity

Implicativity and factivity are concerned with analyzing the truth conditions of nested propositions (Nairn et al., 2006). For example, in the premise of the entailment pair shown in example (6) below, the *locking* is the event that Ed *forgot to* do, meaning that it did not happen. In example (7), *build* is the main verb of the complement of *hope*, so we cannot infer that the *building* event occurred, nor can we infer that it did not occur. Correctly recognizing nested propositions and analyzing their contexts is necessary for preventing the licensing of entailments like (6) and rejecting those like (7).

(6) *p:* Ed forgot to lock the door.[3]
h: Ed did not lock the door.

(7) *p:* The mayor hoped to build a new stadium.[4]
h: *The mayor built a new stadium.

Nairn et al. (2006) presented an approach to the treatment of inferences involving implicatives and factives. Their approach identifies an "implication signature" for every implicative or factive verb. This signature specifies the truth conditions for the verb's nested proposition, depending on whether the verb occurs in a positive or negative environment. Following MacCartney and Manning (2009), we write implication signatures as "x/y" where x represents the entailment to which the speaker commits in a positive environment and y represents entailment in a negative environment. Both x and y have three possible values: "+" for positive entailment, meaning the nested proposition is entailed, "-" for negative entailment, meaning the negation of the proposition is entailed, and "o" for "null" entailment, meaning that neither the proposition nor its negation is entailed. Figure 2 gives concrete examples.[5]

	signature	example
forgot that	+/+	he forgot that Dave left $\models$ Dave left
		he did not forget that Dave left $\models$ Dave left
managed to	+/-	he managed to escape $\models$ he escaped
		he did not manage to escape $\models$ he did not escape
forgot to	-/+	he forgot to pay $\models$ he did not pay
		he did not forget to pay $\models$ he paid
refused to	-/o	he refused to fight $\models$ he did not fight
		he did not refuse to fight $\not\models$ {he fought, he did not fight}

Fig. 2: Implication Signatures

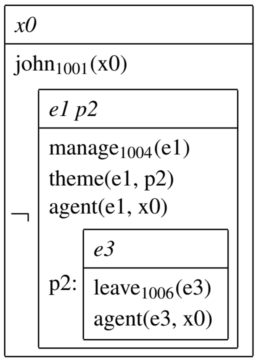
[3] Example (6) is derived from examples by MacCartney and Manning (2009).

[4] Example (7) is adapted from document wsj_0126 from the Penn Treebank.

[5] Note that *forget to* and *forget that* have different implication signatures. As such, in order to select the right signature, it is necessary to examine not simply the verb but the entire subcategorization frame. To do this, we make use of the dependency parse generated by the C&C parser that is input to Boxer.

Inferences with nested propositions

The standard conversion from DRT to first-order logic (FOL) (the one used by Boxer) falls short in its analysis of nested propositions. Consider the entailment pair “John did not manage to leave” and “John left”. The DRT interpretation of the premise and its corresponding FOL conversion are shown in Figure 3.



(a) DRT interpretation

$$\begin{aligned} &\exists\, x0.[\; john_{1001}(x0)\ \& \\ &\quad \neg\exists\, e1 p2.[\; manage_{1004}(e1)\ \& \\ &\qquad theme(e1,p2)\ \& \\ &\qquad agent(e1,x0)\ \& \\ &\qquad \exists\, e3.[\; leave_{1006}(e3)\ \& \\ &\qquad\quad agent(e3,x0)]]] \end{aligned}$$

(b) FOL translation

Fig. 3: Boxer’s DRT interpretation of “John did not manage to leave.”

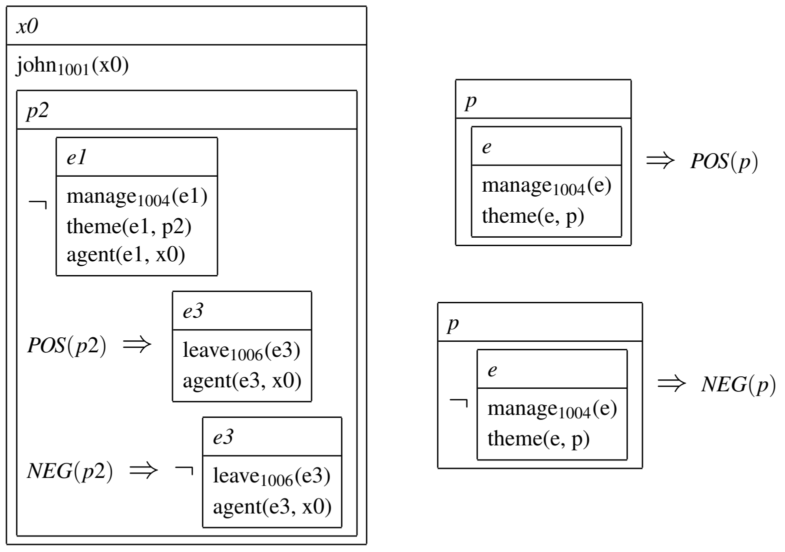
It should be clear that “John did not manage to leave” does *not* entail “John left” (and, in fact, entails the opposite). Unfortunately, the FOL formula shown in Figure 3b *does* entail the FOL representation of “John left”, which is

$$\exists\, x0\ e1.[john_{1001}(x0)\ \&\ leave_{1006}(e1)\ \&\ agent(e1,x0)]$$

The incorrect inference occurs here because the standard DRT-to-FOL translation loses some information. DRT expressions are allowed to have *labeled subexpressions*, such as $p2$ in Figure 3a that is used to reference the *theme* of the *manage* event: the *leave* event. The FOL expression, on the other hand, shows that $p2$ is the theme of event $e1$, but has no way of stating what $p2$ refers to.

In order to capture the information that the DRT labels provide, we modify the DRT expression to contain explicit *subexpression triggers*. That is, for a sub-DRS A labeled by p, we replace A with two new expressions in the same scope: $POS(p) \rightarrow A$ and $NEG(p) \rightarrow \neg A$. The result of such a replacement on the DRS in Figure 3a can be see in Figure 4a.

Now that our labeled subexpression has triggers, we can introduce inference rules to activate those triggers. The purpose of these inference rules is to capture the behavior dictated by the implication signature of the implicative or factive verb for which the relevant subexpression is the theme. For example, according to the implication



(a) DRT interpretation with subexpression triggers

(b) Subexpression-triggering inference rules for implicative "manage to" with signature +/-

Fig. 4: First (insufficient) attempt at correcting for the loss of labeled sub-expression information.

signature in Figure 2, the implicative *manage to* is positively entailing in positive contexts and negatively entailing in negative contexts. This means that if John *managed to* do what is described by *p*, then the event described by *p* occurred, or in other words, the subexpression of *p* is *true*. Likewise, if John *did not manage to* do what is described by *p*, then the event described by *p did not* occur, meaning that the subexpression of *p* is *false*.

The triggering inference rules for *managed to* are shown in Figure 4b. The first rule, for positive contexts, says that for all propositions *p*, if *p* is "managed", then *p*'s subexpression is *true*, so trigger the "positive entailment" subexpression which, in our example, says that the *leaving* event occurred. The second rule, for negative contexts, says that for all propositions *p*, if there is *no* "managing" of *p*, then *p*'s subexpression is *false*, so trigger the "negative entailment" subexpression to say that there is *no* event of *leaving*.

While this approach works for positive contexts, there is a subtle problem for negative contexts. The negative context rule in Figure 4b can be translated to FOL as

$$\forall\, p.[\ \neg\exists\ e.[\ manage_{1004}(e) \wedge theme(e,p))\ ] \rightarrow NEG(p)\ ]$$

This expression is stating that for all propositions p, p is *false* if there is no "managing" of p. Now, we want this inference rule to be used in cases where it is stated that "managing" did not occur, such as in the expression of Figure 4a, where we see that is is the case that

$$\neg \begin{array}{|l|}\hline \textit{e1} \\ \hline \text{manage}_{1004}\text{(e1)} \\ \text{theme(e1, p2)} \\ \text{agent(e1, x0)} \\ \hline \end{array}$$

which is equivalent to the FOL expression

$$\neg\exists\, e1\ [\ manage_{1004}(e1) \wedge theme(e1,p2) \wedge agent(e1,x0)\ ]$$

stating that there is no "managing" of $p2$ by $x0$. However, the antecedent of our negative context rule states that there is *no* "managing" of the proposition, so the rule would only be used if it could be proven that there is no "managing" event at all. Unfortunately, stating that $p2$ is not "managed" *by x0* does *not* entail that $p2$ is not "managed" at all since $p2$ could be managed by someone other than $x0$.

To overcome this problem, we modify our representation of a negated event. Instead of representing an event, such as the "managing" event, that *did not* occur as $\neg\exists\ e.[\ manage(e)\ ]$, we represent it explicitly as an event of *non-occurrence*: $\exists\ e.[\ \textbf{not}\_manage(e)\ ]$. Applying this change to the DRS and inference rules in Figure 4, we arrive at our final form in Figure 5.

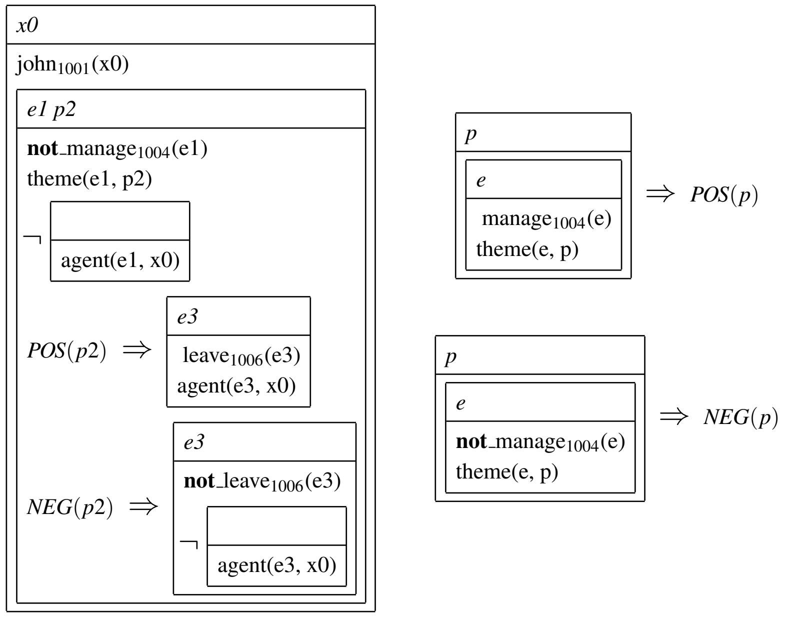
Using this strategy, we can see that the negative context rule is active when there exists a "not-managing" state, and the representation of "John did not manage to leave" explicitly requires that there is such an state, meaning that the rule will be used in the inference. With all of these pieces in place, the inference works as expected.

Thus, we transform the output of Boxer in two ways. First, we identify any labeled propositions and replace them with pairs of proposition triggers. Then, we modify any negated DRSs by extracting the verb and theme atoms, changing the verb predicate to a "not_" predicate[6], and finally ensuring that all other expressions under the negated DRS (aside from the labeled proposition itself), remain under a negated DRS.

Once the sentence representations have been modified, we generate inference rules for each implicative verb. If the verb is positively entailing in positive contexts, we generate a rule of the form

$$\forall\ p.[\ \exists\ e.[\ \langle verb\rangle(e) \wedge theme(e,p))\ ] \rightarrow POS(p)\ ]$$

[6] The lexical mapping for these new predicates ignores the negation, i.e. $\ell(\text{not\_manage}) = \ell(\text{manage})$.



(a) DRT interpretation with subexpression triggers

(b) Subexpression-triggering inference rules for implicative "manage to" with signature +/-

Fig. 5: Explicit capturing of sub-expression information.

but if it is negatively entailing in positive contexts, we instead generate a rule of the form

$$\forall\, p.[\ \exists\, e.[\ \langle verb\rangle(e) \wedge theme(e,p))\ ] \rightarrow NEG(p)\ ]$$

If the verb is positively entailing in *negative* contexts, we generate a rule of the form

$$\forall\, p.[\ \exists\, e.[\ \textbf{not}\_\langle verb\rangle(e) \wedge theme(e,p))\ ] \rightarrow POS(p)\ ]$$

but if it is negatively entailing in negative contexts, we instead generate a rule of the form

$$\forall\, p.[\ \exists\, e.[\ \textbf{not}\_\langle verb\rangle(e) \wedge theme(e,p))\ ] \rightarrow NEG(p)\ ]$$

If the verb is non-entailing in either positive or negative contexts, then we do not generate a rule for that context polarity.

This approach works for arbitrarily long chains of nested implicatives and factives. For example, consider the entailment in (8).

(8) Dave managed to fail to not forget to leave ⊨ Dave did not leave

Our approach is able to predict this entailment by correctly handling the three nested implicatives along with the negation. Figure 6 shows the nested polarity environments and how the implicative verbs and negations modify the polarity. The top-level verb *managed to* maintains its same polarity and predicts a positive environment for the *fail to* event. The *fail to* reverses the polarity for the *not forget to* state. Since the negation of *forget* is in a negative environment, the negations cancel, putting *forget* in a positive environment, thus predicting a negative environment for the *leaving* event. Since the *leaving* event is in a negative environment, we can say that the sentence entails that the leaving did not occur.

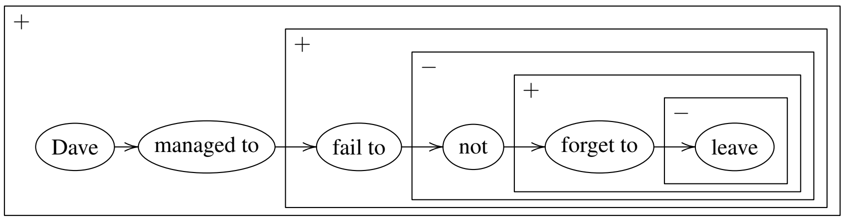


Fig. 6: Nested polarity environments showing how implicative verbs and negation modify polarity.

Interaction with other phenomena

MacCartney and Manning (2009) extended the work by Nairn et al. (2006) in order to correctly treat inference involving monotonicity and exclusion. Our approach to implicativity and factivity combines naturally with hypernymy to ensure correct entailment judgements. For example, no additional work is required to license the entailments in (9).

(9) (a) John refused to dance ⊨ John didn't tango

(b) John did not forget to tango ⊨ John danced

Likewise, no further work is needed for our implicativity and factivity approach to interact correctly with our approach to ambiguity in word meaning. For example, consider example (10). Here the premise contains the verb *prevent* in a positive context, which is negatively entailing. It also contains the word *leave* which is synonymous with both *result in* and *flee* through different senses. As the example shows, our approach is able to correctly handle the interaction between the lexical ambiguity and the implicative verb.

(10) *p:* He prevented the spill from leaving a stain.

h1: The spill did not result in a stain.

h2:* The spill did not flee a stain.

h3:* The spill resulted in a stain.

In example (11), the *prevent* event is nested under the null-entailing verb *try*. As such, neither alternate sense of *leave* is entailed since *try* says nothing about the truth or falsity of its nested proposition.

(11) *p:* He tried to prevent the spill from leaving a stain.

h1:* The spill did not result in a stain.

h2:* The spill did not flee a stain.

h3:* The spill resulted in a stain.

7 Preliminary Evaluation

As a preliminary evaluation of our system, we constructed the set of demonstrative examples included in this paper to test our system's ability to handle the previously discussed phenomena and their interactions. We ran each example with both a standard first-order theorem prover and Alchemy to ensure that the examples work as expected. Note that since weights are not possible when running an example in the theorem prover, any rule that would receive a non-zero weight in an MLN is simply treated as a "hard clause" following Bos and Markert (2005). For the experiments, we generated a vector space from the entire New York Times portion of the English Gigaword corpus (Graff and Cieri, 2003).

The example entailments evaluated were designed to test the interaction between the logical and weighted phenomena. For example, in (12), "fail to" is a negatively entailing implicative in a positive environment, so according to the theorem prover, *p* entails both *h1* and *h2*. However, using our weighted approach, Alchemy outputs that *h1* is more probable than *h2*.

(12) *p:* The U.S. is watching closely as South Korea fails to honor U.S. patents[7]

h1: South Korea does not **observe** U.S. patents

h2:* South Korea does not **reward** U.S. patents

The first-order approach, which contains inference rules for both paraphrases as hard clauses, cannot distinguish between good and bad paraphrases, and considers both of them equally valid. In contrast, the weighted approach can judge the degree

[7] Sentence adapted from Penn Treebank document wsj_0020.

of fit of the two potential paraphrases. Also, it does so in a context-specific manner, choosing the paraphrase *observe* over *reward* in the context of *patents*.

Our ability to perform a full-scale evaluation is currently limited by problems in the Alchemy software required to perform probabilistic inference. This is discussed more in Section 8.

8 Future work

Our plans for continued work can be divided into two categories: work on the theoretical side and work on implementation and evaluation.

From a theoretical perspective, we have used a simplistic bag-of-words approach for computing a context-specific vector for a predicate based on its formula context (functions α and κ). We plan to move to a more informative construction that takes semantic relations into account. This will be interesting in particular because the relations that can be read off a logical form differ from those available in a dependency parse. For example, we can check whether two predicates occur within the same DRS, or whether they apply to a common variable. We can also ask what influence different logical connectives have on perceived word meaning.

Additionally, up to this point we have only addressed word-level paraphrasing with weighted lexical ambiguity rules that connect individual words. However, our framework could easily be extended to allow for weighted paraphrase rules for higher-order phrases such as noun-noun compounds, adjective-noun compounds, or full noun phrases.

We would also like to extend our formalism to address a wider range of linguistic phenomena. Many phenomena are better described using weights than through categorial analyses, and first-order representations do not correctly address this. By extending our framework, we hope to be able to apply weights derived from distributional information to a wide variety of modeled concepts. The inference rules generated by our approach to factivity might be good candidates for this extension. Nairn et al. (2006) proposed that there may be “degrees of factivity” based on the context of the verb. Because the inference rules that we use to activate the presupposition triggers are externalized, they can be weighted independently of the rest of the semantic analysis. Right now the rules are either generated or not, which is equivalent to assigning a weight of either 1 or 0, but a weighted approach could be taken instead.

From an implementation perspective, we would like to run a large-scale evaluation of our techniques. However, the major barrier to scaling up is that the Alchemy software has severe inefficiencies in terms of memory requirements and speed. This prevents us from executing larger and more complex examples. There is on-going work to improve Alchemy (Gogate and Domingos, 2011), so we hope to be able to make use of new probabilistic inference tools as they become available.

9 Conclusion

In this paper, we have defined a link between logical form and vector spaces through a lexical mapping of predicate symbols to points in space. We address polysemy not through separate predicate symbols for different senses of a word, but by using a single predicate symbol with a lexical mapping that gets adapted to the context in which the predicate symbol appears. We use the link to project weighted inferences from the vector space to the logical form.

We showed how these weighted first-order representations can be used to perform probabilistic first-order inferences using Markov Logic. We have shown how our approach handles three distinct phenomena, word meaning ambiguity, hypernymy, and implicativity, as well as allowing them to interact appropriately. Most importantly our approach allows us to model some phenomena with hard first-order techniques and other phenomena with soft weights, and to do all of this within a single, unified framework. The resulting approach is able to correctly solve a number of difficult textual entailment problems that require handling complex combinations of these important semantic phenomena.

Acknowledgements

This work was supported by the U.S. Department of Defense through a National Defense Science and Engineering Graduate Fellowship (NDSEG) Fellowship for the first author, National Science Foundation grant IIS-0845925 for the second author, Air Force Contract FA8750-09-C-0172 under the DARPA Machine Reading Program and NSF grants IIS-0712097 and IIS-1016312 for the third author, and a grant from the Texas Advanced Computing Center (TACC), The University of Texas at Austin.

We would like to thank Gemma Boleda, Louise McNally, Karl Pichotta, and Yinon Bentor for their helpful comments.

References

Bos, J., S. Clark, M. Steedman, J. R. Curran, and J. Hockenmaier (2004). Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING 2004*, Geneva, Switzerland, pp. 1240–1246.

Bos, J. and K. Markert (2005). Recognising textual entailment with logical inference. In *Proceedings of EMNLP 2005*, Vancouver, B.C., Canada, pp. 628–635.

Clark, S. and J. R. Curran (2004). Parsing the WSJ using CCG and log-linear models. In *Proceedings of ACL 2004*, Barcelona, Spain, pp. 104–111.

Dagan, I., O. Glickman, and B. Magnini (2005). The PASCAL Recognising Textual Entailment challenge. In *In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Dinu, G. and M. Lapata (2010). Measuring distributional similarity in context. In *Proceedings of EMNLP 2010*, Cambridge, MA, pp. 1162–1172.

Erk, K. (2010). What is word meaning, really? (and how can distributional models help us describe it?). In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, Uppsala, Sweden, pp. 17–26.

Erk, K. and S. Padó (2008). A structured vector space model for word meaning in context. In *Proceedings of EMNLP 2008*, Honolulu, HI, pp. 897–906.

Erk, K. and S. Padó (2010). Exemplar-based models for word meaning in context. In *Proceedings of ACL 2010*, Uppsala, Sweden, pp. 92–97.

Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In *Studies in linguistic analysis*, pp. 1–32. Oxford, England: Blackwell Publishers.

Gärdenfors, P. (2004). *Conceptual spaces*. Cambridge, MA: MIT press.

Getoor, L. and B. Taskar (Eds.) (2007). *Introduction to Statistical Relational Learning*. Cambridge, MA: MIT Press.

Gogate, V. and P. Domingos (2011). Probabilistic theorem proving. In *In 27th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Graff, D. and C. Cieri (2003). English Gigaword. Linguistic Data Consortium, Philadelphia.

Harris, Z. (1954). Distributional structure. *Word 10*, 146–162.

Hobbs, J. R., M. Stickel, D. Appelt, and P. Martin (1993). Interpretation as abduction. *Artificial Intelligence 63*(1–2), 69–142.

Kamp, H. and U. Reyle (1993). *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Dordrecht: Kluwer.

Kok, S., P. Singla, M. Richardson, and P. Domingos (2005). The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington. `http://www.cs.washington.edu/ai/alchemy`.

Kotlerman, L., I. Dagan, I. Szpektor, and M. Zhitomirsky-Geffet (2010). Directional distributional similarity for lexical inference. *Natural Language Engineering 16*(04), 359–389.

Krishnamurthy, R. and D. Nicholls (2000). Peeling an onion: the lexicographers' experience of manual sense-tagging. *Computers and the Humanities 34*(1–2).

Landauer, T. and S. Dumais (1997). A solution to Platos problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review 104*(2), 211–240.

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

Lund, K. and C. Burgess (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers 28*, 203–208.

MacCartney, B. and C. D. Manning (2009). An extended model of natural logic. In *Proceedings of the Eighth International Conference on Computational Semantics (IWCS-8)*, pp. 140–156.

Miller, G. A. (2009). Wordnet - about us. `http://wordnet.princeton.edu`.

Mitchell, J. and M. Lapata (2008). Vector-based models of semantic composition. In *Proceedings of ACL*, pp. 236–244.

Montague, R. (1970). Universal grammar. *Theoria 36*, 373–398. Reprinted in Thomason (1974), pp 7-27.

Nairn, R., C. Condoravdi, and L. Karttunen (2006). Computing relative polarity for textual inference. In *Proceedings of Inference in Computational Semantics (ICoS-5)*, Buxton, UK.

Poon, H. and P. Domingos (2009). Unsupervised semantic parsing. In *Proceedings of EMNLP 2009*, pp. 1–10.

Reisinger, J. and R. J. Mooney (2010). Multi-prototype vector-space models of word meaning. In *Proceedings of NAACL 2010*.

Richardson, M. and P. Domingos (2006). Markov logic networks. *Machine Learning 62*, 107–136.

Szpektor, I., I. Dagan, R. Bar-Haim, and J. Goldberger (2008). Contextual preferences. In *Proceedings of ACL 2008*, Columbus, OH, pp. 683–691.

Thater, S., H. Fürstenau, and M. Pinkal (2010). Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of ACL 2010*, Uppsala, Sweden, pp. 948–957.

Thomason, R. H. (Ed.) (1974). *Formal Philosophy. Selected Papers of Richard Montague*. New Haven: Yale University Press.

Van de Cruys, T., T. Poibeau, and A. Korhonen (2011). Latent vector weighting for word meaning in context. In *Proceedings of EMNLP 2011*, Edinburgh, Scotland, pp. 1012–1022.