Logical Inference on Dependency-based Compositional Semantics

Ran Tian **Yusuke Miyao** **Takuya Matsuzaki**
National Institute of Informatics, Japan
{tianran,yusuke,takuya-matsuzaki}@nii.ac.jp

Abstract

Dependency-based Compositional Semantics (DCS) is a framework of natural language semantics with easy-to-process structures as well as strict semantics. In this paper, we equip the DCS framework with logical inference, by defining *abstract denotations* as an abstraction of the computing process of denotations in original DCS. An inference engine is built to achieve inference on abstract denotations. Furthermore, we propose a way to generate on-the-fly knowledge in logical inference, by combining our framework with the idea of tree transformation. Experiments on FraCaS and PASCAL RTE datasets show promising results.

1 Introduction

Dependency-based Compositional Semantics (DCS) provides an intuitive way to model semantics of questions, by using simple dependency-like trees (Liang et al., 2011). It is expressive enough to represent complex natural language queries on a relational database, yet simple enough to be latently learned from question-answer pairs. In this paper, we equip DCS with *logical inference*, which, in one point of view, is "the best way of testing an NLP system's semantic capacity" (Cooper et al., 1996).

It should be noted that, however, a framework primarily designed for question answering is not readily suited for logical inference. Because, answers returned by a query depend on the specific database, but implication is independent of any databases. For example, answers to the question "*What books are read by students?*", should always be a subset of answers to "*What books are ever read by anyone?*", no matter how we store the data of students and how many records of books are there in our database.

Thus, our first step is to fix a notation which abstracts the calculation process of DCS trees, so as to clarify its meaning without the aid of any existing database. The idea is to borrow a minimal set of operators from relational algebra (Codd, 1970), which is already able to formulate the calculation in DCS and define *abstract denotation*, which is an abstraction of the *computation* of denotations guided by DCS trees. Meanings of sentences then can be represented by primary relations among abstract denotations. This formulation keeps the simpleness and computability of DCS trees mostly unaffected; for example, our semantic calculation for DCS trees is parallel to the denotation computation in original DCS.

An inference engine is built to handle inference on abstract denotations. Moreover, to compensate the lack of background knowledge in practical inference, we combine our framework with the idea of tree transformation (Bar-Haim et al., 2007), to propose a way of generating knowledge in logical representation from entailment rules (Szpektor et al., 2007), which are by now typically considered as syntactic rewriting rules.

We test our system on FraCaS (Cooper et al., 1996) and PASCAL RTE datasets (Dagan et al., 2006). The experiments show: (i) a competitive performance on FraCaS dataset; (ii) a big impact of our automatically generated on-the-fly knowledge in achieving high recall for a logic-based RTE system; and (iii) a result that outperforms state-of-the-art RTE system on RTE5 data. Our whole system is publicly released and can be downloaded from `http://kmcs.nii.ac.jp/tianran/tifmo/`.

2 The Idea

In this section we describe the idea of representing natural language semantics by DCS trees, and achieving inference by computing logical relations among the corresponding abstract denotations.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 79–89,
Baltimore, Maryland, USA, June 23-25 2014.

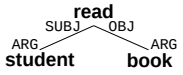

Figure 1: The DCS tree of "*students read books*"

student

ARG
Mark
John
Emily
...

book

ARG
A Tale of Two Cities
Ulysses
...

read

SUBJ	OBJ
Mark	New York Times
Mary	A Tale of Two Cities
John	Ulysses
...	...

Table 1: Databases of *student*, *book*, and *read*

2.1 DCS trees

DCS trees has been proposed to represent natural language semantics with a structure similar to dependency trees (Liang et al., 2011) (Figure 1). For the sentence "*students read books*", imagine a database consists of three tables, namely, a set of students, a set of books, and a set of "reading" events (Table 1). The DCS tree in Figure 1 is interpreted as a command for querying these tables, obtaining "reading" entries whose "SUBJ" field is **student** and whose "OBJ" field is **book**. The result is a set $\{$*John reads Ulysses*, ...$\}$, which is called a *denotation*.

DCS trees can be extended to represent linguistic phenomena such as quantification and coreference, with additional markers introducing additional operations on tables. Figure 2 shows an example with a quantifier "*every*", which is marked as "$\subset$" on the edge (**love**)OBJ−ARG(**dog**) and interpreted as a *division operator* $q_{\subset}^{\text{OBJ}}$ (§2.2). Optimistically, we believe DCS can provide a framework of semantic representation with sufficiently wide coverage for real-world texts.

The strict semantics of DCS trees brings us the idea of applying DCS to logical inference. This is not trivial, however, because DCS works under the assumption that databases are explicitly available. Obviously this is unrealistic for logical inference on unrestricted texts, because we cannot prepare a database for everything in the world. This fact fairly restricts the applicable tasks of DCS.

Our solution is to redefine DCS trees without the aid of any databases, by considering each node of a DCS tree as a content word in a sentence (but may no longer be a table in a specific database), while each edge represents semantic relations between two words. The labels on both ends of an edge, such as SUBJ (subject) and OBJ (object), are considered as *semantic roles* of the corresponding words[1]. To formulate the database querying process defined by a DCS tree, we provide formal semantics to DCS trees by employing *relational algebra* (Codd, 1970) for representing the query. As described below, we represent meanings of sentences with *abstract denotations*, and logical relations among sentences are computed as relations among their abstract denotations. In this way, we can perform inference over formulas of relational algebra, without computing database entries explicitly.

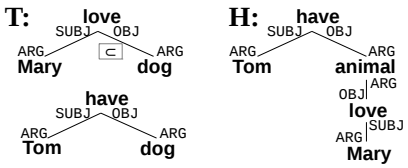

Figure 2: DCS trees of "*Mary loves every dog*" (Left-Up), "*Tom has a dog*" (Left-Down), and "*Tom has an animal that Mary loves*" (Right).

2.2 Abstract denotations

Abstract denotations are formulas constructed from a minimal set of relational algebra (Codd, 1970) operators, which is already able to formulate the database queries defined by DCS trees.

For example, the semantics of "*students read books*" is given by the abstract denotation:

$$F_1 = \textbf{read} \cap (\textbf{student}_{\text{SUBJ}} \times \textbf{book}_{\text{OBJ}}),$$

where **read**, **student** and **book** denote sets represented by these words respectively, and w_r represents the set w considered as the domain of the semantic role r (e.g. $\textbf{book}_{\text{OBJ}}$ is the set of books considered as objects). The operators $\cap$ and $\times$ represent intersection and Cartesian product respectively, both borrowed from relational algebra. It is not hard to see the abstract denotation denotes the intersection of the "*reading*" set (as illustrated by the "*read*" table in Table 1) with the product of "*student*" set and "*book*" set, which results in the same denotation as computed by the DCS tree in Figure 1, i.e. $\{$*John reads Ulysses*, ...$\}$. However, the point is that F_1 itself is an algebraic formula that does not depend on any concrete databases.

Formally, we introduce the following *constants*:

- W: a universal set containing all entities.

[1]The semantic role ARG is specifically defined for denoting nominal predicate.

	example phrase	abstract denotation / statement
compound noun	*pet fish*	**pet** $\cap$ **fish**
modification	*nice day*	**day** $\cap$ ($W_{\text{ARG}} \times$ **nice**$_{\text{MOD}}$)
temporal relation	*boys study at night*	**study** $\cap$ (**boy**$_{\text{SUBJ}} \times$ **night**$_{\text{TIME}}$)
relative clause	*books that students read*	**book** $\cap\, \pi_{\text{OBJ}}$(**read** $\cap$(**student**$_{\text{SUBJ}} \times W_{\text{OBJ}}$))
quantification	*all men die*	**man** $\subset\, \pi_{\text{SUBJ}}$(**die**)
hypernym		**dog** $\subset$ **animal**
derivation	*all criminals commit a crime*	**criminal** $\subset\, \pi_{\text{SUBJ}}$(**commit**$\cap$ ($W_{\text{SUBJ}} \times$ **crime**$_{\text{OBJ}}$))
antonym		**rise** $\|$ **fall**
negation	*no dogs are hurt*	**dog** $\|\, \pi_{\text{OBJ}}$(**hurt**)

Table 2: Abstract denotations and statements

- Content words: a content word (e.g. *read*) defines a set representing the word (e.g. **read** $= \{(x, y) \mid read(x, y)\}$).

In addition we introduce following *functions*:

- $\times$: the Cartesian product of two sets.
- $\cap$: the intersection of two sets.
- π_r: projection onto domain of semantic role r (e.g. $\pi_{\text{OBJ}}(\textbf{read}) = \{y \mid \exists x; read(x, y)\}$). Generally we admit projections onto multiple semantics roles, denoted by π_R where R is a set of semantic roles.
- ι_r: relabeling (e.g. $\iota_{\text{OBJ}}(\textbf{book}) = \textbf{book}_{\text{OBJ}}$).
- $q^r_{\subset}$: the division operator, where $q^r_{\subset}(A, B)$ is defined as the largest set X which satisfies $B_r \times X \subset A$.[2] This is used to formulate universal quantifiers, such as "*Mary loves every dog*" and "*books read by all students*".

An *abstract denotation* is then defined as finite applications of functions on either constants or other abstract denotations.

2.3 Statements

As the semantics of DCS trees is formulated by abstract denotations, the meanings of declarative sentences are represented by *statements* on abstract denotations. Statements are declarations of some relations among abstract denotations, for which we consider the following set relations:

Non-emptiness $A \neq \emptyset$: the set A is not empty.
Subsumption $A \subset B$: set A is subsumed by B.[3]

Roughly speaking, the relations correspond to the logical concepts *satisfiability* and *entailment*.

[2] If A and B has the same dimension, $q_{\subset}(A, B)$ is either $\emptyset$ or $\{*\}$ (0-dimension point set), depending on if $A \subset B$.

[3] Using division operator, subsumption can be represented by non-emptiness, since for sets A, B of the same dimension, $q_{\subset}(A, B) \neq \emptyset \Leftrightarrow A \subset B$.

Abstract denotations and statements are convenient for representing semantics of various types of expressions and linguistic knowledge. Some examples are shown in Table 2.[4]

2.4 Logical inference on DCS

Based on abstract denotations, we briefly describe our process to apply DCS to textual inference.

2.4.1 Natural language to DCS trees

To obtain DCS trees from natural language, we use Stanford CoreNLP[5] for dependency parsing (Socher et al., 2013), and convert Stanford dependencies to DCS trees by pattern matching on POS tags and dependency labels.[6] Currently we use the following semantic roles: `ARG`, `SUBJ`, `OBJ`, `IOBJ`, `TIME` and `MOD`. The semantic role `MOD` is used for any restrictive modifiers. Determiners such as "all", "every" and "each" trigger quantifiers, as shown in Figure 2.

2.4.2 DCS trees to statements

A DCS tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ is defined as a rooted tree, where each node $\sigma \in \mathcal{N}$ is labeled with a content word $w(\sigma)$ and each edge $(\sigma, \sigma') \in \mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ is labeled with a pair of semantic roles (r, r')[7]. Here σ is the node nearer to the root. Furthermore, for each edge (σ, σ') we can optionally assign a quantification marker.

Abstract denotation of a DCS tree can be calculated in a bottom-up manner. For example, the abstract denotation of **H** in Figure 2 is calculated from the leaf node **Mary**, and then:

Node **love** (*Mary loves*):
$F_2 = \textbf{love} \cap (\textbf{Mary}_{\text{SUBJ}} \times W_{\text{OBJ}})$
Node **animal** (*Animal that Mary loves*):
$F_3 = \textbf{animal} \cap \pi_{\text{OBJ}}(F_2)$
Node **have** (*Tom has an animal that Mary loves*):
$F_4 = \textbf{have} \cap (\textbf{Tom}_{\text{SUBJ}} \times (F_3)_{\text{OBJ}})$.

Formally, suppose the root σ of a DCS tree $\mathcal{T}$ has children $\tau_1, \ldots, \tau_n$, and edges $(\sigma, \tau_1), \ldots, (\sigma, \tau_n)$ labeled by $(r_1, r'_1), \ldots, (r_n, r'_n)$, respectively. The abstract denotation of $\mathcal{T}$ is defined as:

$$[\![\mathcal{T}]\!] = w(\sigma) \cap (\bigcap_{i=1}^{n} \iota_{r_i}(\pi_{r'_i}([\![\mathcal{T}_{\tau_i}]\!])) \times W_{R_\sigma \setminus r_i}),$$

[4] Negation and disjointness ("$\|$") are explained in §2.5.

[5] `http://nlp.stanford.edu/software/corenlp.shtml`

[6] In (Liang et al., 2011) DCS trees are learned from QA pairs and database entries. We obtain DCS trees from dependency trees, to bypass the need of a concrete database.

[7] The definition differs slightly from the original Liang et al. (2011), mainly for the sake of simplicity and clarity.

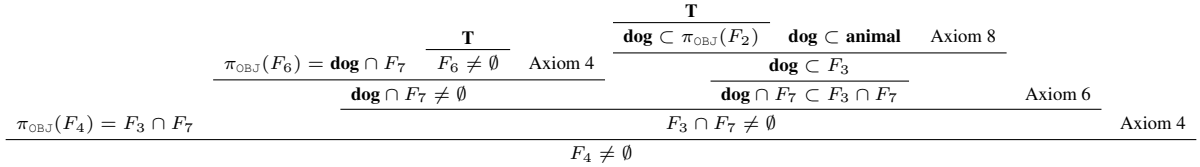

Figure 3: An example of proof using abstract denotations

1. $W \neq \emptyset$	5. $(A \subset B \;\&\; B \subset C) \Rightarrow A \subset C$
2. $A \cap B \subset A$	6. $(A \subset B \;\&\; A \neq \emptyset) \Rightarrow B \neq \emptyset$
3. $B_r \times q^r_{\subset}(A, B) \subset A$	7. $A \subset B \Rightarrow \pi_R(A) \subset \pi_R(B)$
4. $\pi_R(A) \neq \emptyset \Leftrightarrow A \neq \emptyset$	8. $(C \subset A \;\&\; C \subset B) \Rightarrow C \subset A \cap B$

Table 3: An excerpt of axioms

where $\mathcal{T}_{\tau_i}$ is the subtree of $\mathcal{T}$ rooted at τ_i, and R_σ is the set of possible semantic roles for content word $w(\sigma)$ (e.g. $R_{\textbf{love}} = \{\texttt{SUBJ}, \texttt{OBJ}\}$), and $W_{R_\sigma \setminus r_i}$ is the product of W which has dimension $R_\sigma \setminus r_i$ (e.g. $W_{\{\texttt{SUBJ},\texttt{OBJ}\}\setminus\texttt{SUBJ}} = W_{\texttt{OBJ}}$).

When universal quantifiers are involved, we need to add division operators to the formula. If (σ, τ_i) is assigned by a quantification marker "$\subset$"[8], then the abstract denotation is[9]

$$[\![\mathcal{T}]\!] = q^{r_i}_{\subset}(\pi_{R_\sigma \setminus \{r_1,\ldots,r_{i-1}\}}([\![\mathcal{T}']\!]), \pi_{r'_i}([\![\mathcal{T}_{\tau_i}]\!])),$$

where $\mathcal{T}'$ is the same tree as $\mathcal{T}$ except that the edge (σ, τ_i) is removed. For example, the abstract denotation of the first sentence of **T** in Figure 2 (*Mary loves every dog*) is calculated from F_2 (*Mary loves*) as

$$F_5 = q^{\texttt{OBJ}}_{\subset}(\pi_{\texttt{OBJ}}(F_2), \textbf{dog}).$$

After the abstract denotation $[\![\mathcal{T}]\!]$ is calculated, the statement representing the meaning of the sentence is defined as $[\![\mathcal{T}]\!] \neq \emptyset$. For example, the statement of "*students read books*" is $\textbf{read} \cap (\textbf{student}_{\texttt{SUBJ}} \times \textbf{book}_{\texttt{OBJ}}) \neq \emptyset$, and the statement of "*Mary loves every dog*" is $q^{\texttt{OBJ}}_{\subset}(\pi_{\texttt{OBJ}}(F_2), \textbf{dog}) \neq \emptyset$, which is logically equivalent to $\textbf{dog} \subset \pi_{\texttt{OBJ}}(F_2)$.[10]

2.4.3 Logical inference

Since meanings of sentences are represented by statements on abstract denotations, logical inference among sentences is reduced to deriving new relations among abstract denotations. This is done by applying axioms to known statements, and approximately 30 axioms are implemented (Table 3).

[8]Multiple quantifiers can be processed similarly.

[9]The result of $[\![\mathcal{T}]\!]$ depends on the order of the children $\tau_1, \ldots, \tau_n$. Different orders correspond to readings of different quantifier scopes.

[10]See Footnote 2,3.

These are algebraic properties of abstract denotations, among which we choose a set of axioms that can be handled efficiently and enable most common types of inference seen in natural language.

For the example in Figure 2, by constructing the following abstract denotations:

Tom has a dog:

$$F_6 = \textbf{have} \cap (\textbf{Tom}_{\texttt{SUBJ}} \times \textbf{dog}_{\texttt{OBJ}})$$

Objects *that Tom has*:

$$F_7 = \pi_{\texttt{OBJ}}(\textbf{have} \cap (\textbf{Tom}_{\texttt{SUBJ}} \times W_{\texttt{OBJ}})),$$

we can use the lexical knowledge $\textbf{dog} \subset \textbf{animal}$, the statements of **T** (i.e. $\textbf{dog} \subset \pi_{\texttt{OBJ}}(F_2)$ and $F_6 \neq \emptyset$), and the axioms in Table 3,[11] to prove the statement of **H** (i.e. $F_4 \neq \emptyset$) (Figure 3).

We built an inference engine to perform logical inference on abstract denotations as above. In this logical system, we treat abstract denotations as *terms* and statements as *atomic sentences*, which are far more easier to handle than first order predicate logic (FOL) formulas. Furthermore, all implemented axioms are horn clauses, hence we can employ forward-chaining, which is very efficient.

2.5 Extensions

Further extensions of our framework are made to deal with additional linguistic phenomena, as briefly explained below.

Negation To deal with negation in our forward-chaining inference engine, we introduce one more relation on abstract denotations, namely *disjointness* $A \parallel B$, meaning that A and B are disjoint sets. Using disjointness we implemented two types of negations: (i) atomic negation, for each content word w we allow negation $\bar{w}$ of that word, characterized by the property $w \parallel \bar{w}$; and (ii) root negation, for a DCS tree $\mathcal{T}$ and its denotation $[\![\mathcal{T}]\!]$, the negation of $\mathcal{T}$ is represented by $\mathcal{T} \parallel \mathcal{T}$, meaning that $\mathcal{T} = \emptyset$ in its effect.

Selection Selection operators in relational algebra select a subset from a set to satisfy some spe-

[11]Algebraic identities, such as $\pi_{\texttt{OBJ}}(F_4) = F_3 \cap F_7$ and $\pi_{\texttt{OBJ}}(F_6) = \textbf{dog} \cap F_7$, are also axioms.

cific properties. This can be employed to represent linguistic phenomena such as downward monotonicity and generalized quantifiers. In the current system, we implement (i) superlatives, e.g. $s_{highest}(\textbf{mountain} \cap (W_{\texttt{ARG}} \times \textbf{Asia}_{\texttt{MOD}}))$ (the highest mountain in Asia) and (ii) numerics, e.g. $s_{two}(\textbf{pet} \cap \textbf{fish})$ (two pet fish), where s_f is a selection marker. Selection operators are implemented as markers assigned to abstract denotations, with specially designed axioms. For example superlatives satisfy the following property: $A \subset B \ \& \ s_{highest}(B) \subset A \Rightarrow s_{highest}(B) = s_{highest}(A)$. New rules can be added if necessary.

Coreference We use Stanford CoreNLP to resolve coreferences (Raghunathan et al., 2010), whereas coreference is implemented as a special type of selection. If a node σ in a DCS tree $\mathcal{T}$ belongs to a mention cluster m, we take the abstract denotation $[\![\mathcal{T}_\sigma]\!]$ and make a selection $s_m([\![\mathcal{T}_\sigma]\!])$, which is regarded as the abstract denotation of that mention. Then all selections of the same mention cluster are declared to be equal.

3 Generating On-the-fly Knowledge

Recognizing textual entailment (RTE) is the task of determining whether a given textual statement **H** can be inferred by a text passage **T**. For this, our primary textual inference system operates as:

1. For a **T**-**H** pair, apply dependency parsing and coreference resolution.
2. Perform rule-based conversion from dependency parses to DCS trees, which are translated to statements on abstract denotations.
3. Use statements of **T** and linguistic knowledge as premises, and try to prove statements of **H** by our inference engine.

However, this method does not work for real-world datasets such as PASCAL RTE (Dagan et al., 2006), because of the knowledge bottleneck: it is often the case that the lack of sufficient linguistic knowledge causes failure of inference, thus the system outputs "no entailment" for almost all pairs (Bos and Markert, 2005).

The transparent syntax-to-semantics interface of DCS enables us to back off to NLP techniques during inference for catching up the lack of knowledge. We extract fragments of DCS trees as paraphrase candidates, translate them back to linguistic expressions, and apply distributional similarity to judge their validity. In this way, our framework combines distributional and logical semantics, which is also the main subject of Lewis and Steedman (2013) and Beltagy et al. (2013).

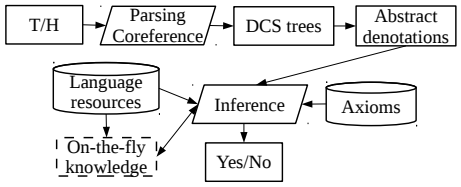

Figure 4: RTE system

As follows, our full system (Figure 4) additionally invokes linguistic knowledge on-the-fly:

4. If **H** is not proven, compare DCS trees of **T** and **H**, and generate path alignments.
5. Aligned paths are evaluated by a similarity score to estimate their likelihood of being paraphrases. Path alignments with scores higher than a threshold are accepted.
6. Convert accepted path alignments into statements on abstract denotations, use them in logical inference as new knowledge, and try to prove **H** again.

3.1 Generating path alignments

On-the-fly knowledge is generated by aligning paths in DCS trees. A path is considered as joining two *germs* in a DCS tree, where a *germ* is defined as a specific semantic role of a node. For example, Figure 5 shows DCS trees of the following sentences (a simplified pair from RTE2-dev):

T: *Tropical storm Debby is blamed for deaths.*
H: *A storm has caused loss of life.*

The germ $\texttt{OBJ}(\textbf{blame})$ and germ $\texttt{ARG}(\textbf{death})$ in DCS tree of **T** are joined by the underscored path. Two paths are aligned if the joined germs are aligned, and we impose constraints on aligned germs to inhibit meaningless alignments, as described below.

3.2 Aligning germs by logical clues

Two germs are aligned if they are both at leaf nodes (e.g. $\texttt{ARG}(\textbf{death})$ in **T** and $\texttt{ARG}(\textbf{life})$ in **H**, Figure 5), or they already have part of their meanings in common, by some logical clues.

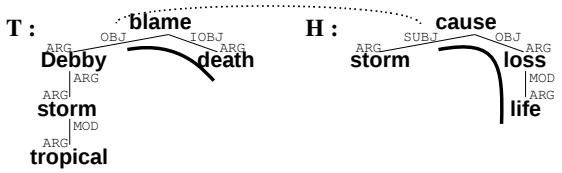

Figure 5: Aligned paths (underscored by the solid lines) and aligned germs (joined by the dotted line)

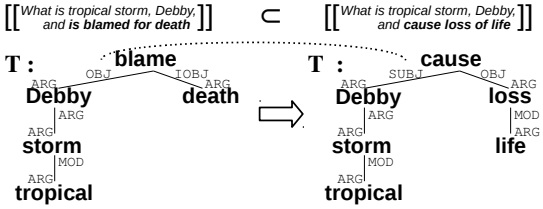

Figure 6: Tree transformation and generated on-the-fly knowledge (subsumption of denotations shown above the trees)

To formulate this properly, we define the abstract denotation of a germ, which, intuitively, represents the meaning of the germ in the specific sentence. The abstract denotation of a germ is defined in a top-down manner: for the root node ρ of a DCS tree $\mathcal{T}$, we define its denotation $[\![\rho]\!]_{\mathcal{T}}$ as the denotation of the entire tree $[\![\mathcal{T}]\!]$; for a non-root node τ and its parent node σ, let the edge (σ, τ) be labeled by semantic roles (r, r'), then define

$$[\![\tau]\!]_{\mathcal{T}} = [\![\mathcal{T}_\tau]\!] \cap (\iota_{r'}(\pi_r([\![\sigma]\!]_{\mathcal{T}})) \times W_{R_\tau \setminus r'}).$$

Now for a germ $r(\sigma)$, the denotation is defined as the projection of the denotation of node σ onto the specific semantic role r: $[\![r(\sigma)]\!]_{\mathcal{T}} = \pi_r([\![\sigma]\!]_{\mathcal{T}})$.

For example, the abstract denotation of germ ARG(**book**) in Figure 1 is defined as $\pi_{\texttt{ARG}}(\textbf{book} \cap \pi_{\texttt{OBJ}}(\textbf{read} \cap (\textbf{student}_{\texttt{SUBJ}} \times \textbf{book}_{\texttt{OBJ}})))$, meaning "*books read by students*". Similarly, denotation of germ OBJ(**blame**) in **T** of Figure 5 indicates the object of "*blame*" as in the sentence "*Tropical storm Debby is blamed for death*", which is a *tropical storm*, is *Debby*, etc. Technically, each germ in a DCS tree indicates a variable when the DCS tree is translated to a FOL formula, and the abstract denotation of the germ corresponds to the set of *consistent values* (Liang et al., 2011) of that variable.

The logical clue to align germs is: if there exists an abstract denotation, other than W, that is a superset of both abstract denotations of two germs, then the two germs can be aligned. A simple example is that ARG(**storm**) in **T** can be aligned to ARG(**storm**) in **H**, because their denotations have a common superset other than W, namely $\pi_{\texttt{ARG}}(\textbf{storm})$. A more complicated example is that OBJ(**blame**) and SUBJ(**cause**) can be aligned, because inference can induce $[\![\texttt{OBJ}(\textbf{blame})]\!]_{\mathbf{T}} = [\![\texttt{ARG}(\textbf{Debby})]\!]_{\mathbf{T}} = [\![\texttt{ARG}(\textbf{storm})]\!]_{\mathbf{T}}$, as well as $[\![\texttt{SUBJ}(\textbf{cause})]\!]_{\mathbf{H}} = [\![\texttt{ARG}(\textbf{storm})]\!]_{\mathbf{H}}$, so they also have the common superset $\pi_{\texttt{ARG}}(\textbf{storm})$. However, for example, logical clues can avoid aligning ARG(**storm**) to ARG(**loss**), which is obviously meaningless.

3.3 Scoring path alignments by similarity

Aligned paths are evaluated by a similarity score, for which we use distributional similarity of the words that appear in the paths (§4.1). Only path alignments with high similarity scores can be accepted. Also, we only accept paths of length ≤ 5, to prevent too long paths to be aligned.

3.4 Applying path alignments

Accepted aligned paths are converted into statements, which are used as new knowledge. The conversion is done by first performing a DCS tree transformation according to the aligned paths, and then declare a subsumption relation between the denotations of aligned germs. For example, to apply the aligned path pair generated in Figure 5, we use it to transform **T** into a new tree **T'** (Figure 6), and then the aligned germs, OBJ(**blame**) in **T** and SUBJ(**cause**) in **T'**, will generate the on-the-fly knowledge: $[\![\texttt{OBJ}(\textbf{blame})]\!]_{\mathbf{T}} \subset [\![\texttt{SUBJ}(\textbf{cause})]\!]_{\mathbf{T'}}$.

Similar to the tree transformation based approach to RTE (Bar-Haim et al., 2007), this process can also utilize lexical-syntactic entailment rules (Szpektor et al., 2007). Furthermore, since the on-the-fly knowledge is generated by transformed pairs of DCS trees, all contexts are preserved: in Figure 6, though the tree transformation can be seen as generated from the entailment rule "*X is blamed for death* → *X causes loss of life*", the generated on-the-fly knowledge, as shown above the trees, only fires with the additional condition that X is a *tropical storm* and is *Debby*. Hence, the process can also be used to generate knowledge from context sensitive rules (Melamud et al., 2013), which are known to have higher quality (Pantel et al., 2007; Clark and Harrison, 2009).

However, it should be noted that using on-the-fly knowledge in logical inference is not a trivial

task. For example, the FOL formula of the rule "*X is blamed for death → X causes loss of life*" is:

$$\forall x; (\exists a; blame(x, a) \ \& \ death(a)) \rightarrow \\ (\exists b, c; cause(x, b) \ \& \ loss(b, c) \ \& \ life(c)),$$

which is not a horn clause. The FOL formula for the context-preserved rule in Figure 6 is even more involved. Still, it can be efficiently treated by our inference engine because as a statement, the formula $[\![\texttt{OBJ}(\textbf{blame})]\!]_{\mathbf{T}} \subset [\![\texttt{SUBJ}(\textbf{cause})]\!]_{\mathbf{T}'}$ is an atomic sentence, more than a horn clause.

4 Experiments

In this section, we evaluate our system on FraCaS (§4.2) and PASCAL RTE datasets (§4.3).

4.1 Language Resources

The lexical knowledge we use are synonyms, hypernyms and antonyms extracted from WordNet[12]. We also add axioms on named entities, stopwords, numerics and superlatives. For example, named entities are singletons, so we add axioms such as $\forall x; (x \subset \textbf{Tom} \ \& \ x \neq \emptyset) \rightarrow \textbf{Tom} \subset x$.

To calculate the similarity scores of path alignments, we use the sum of word vectors of the words from each path, and calculate the cosine similarity. For example, the similarity score of the path alignment "OBJ(**blame**)IOBJ-ARG(**death**) ≈ SUBJ(**cause**)OBJ-ARG(**loss**)MOD-ARG(**life**)" is calculated as the cosine similarity of vectors **blame+death** and **cause+loss+life**. Other structures in the paths, such as semantic roles, are ignored in the calculation. The word vectors we use are from Mikolov et al. (2013)[13] (*Mikolov13*), and additional results are also shown using Turian et al. (2010)[14] (*Turian10*). The threshold for accepted path alignments is set to 0.4, based on preexperiments on RTE development sets.

4.2 Experiments on FraCaS

The FraCaS test suite contains 346 inference problems divided into 9 sections, each focused on a category of semantic phenomena. We use the data by MacCartney and Manning (2007), and experiment on the first section, *Quantifiers*, following Lewis and Steedman (2013). This section has 44 single premise and 30 multi premise problems. Most of the problems do not require lexical knowledge, so we use our primary textual inference system without on-the-fly knowledge nor WordNet, to test the performance of the DCS framework as formal semantics. To obtain the three-valued output (i.e. *yes*, *no*, and *unknown*), we output "*yes*" if **H** is proven, or try to prove the negation of **H** if **H** is not proven. To negate **H**, we use the root negation as described in §2.5. If the negation of **H** is proven, we output "*no*", otherwise we output "*unknown*".

	Single Prem.	Multi Prem.
Lewis13	70	50
MacCartney07	84.1	-
MacCartney08	**97.7**	-
Our Sys.	79.5	**80.0**

Table 4: Accuracy (%) on FraCaS

The result is shown in Table 4. Since our system uses an off-the-shelf dependency parser, and semantic representations are obtained from simple rule-based conversion from dependency trees, there will be only one (right or wrong) interpretation in face of ambiguous sentences. Still, our system outperforms Lewis and Steedman (2013)'s probabilistic CCG-parser. Compared to MacCartney and Manning (2007) and MacCartney and Manning (2008), our system does not need a pretrained alignment model, and it improves by making multi-sentence inferences. To sum up, the result shows that DCS is good at handling universal quantifiers and negations.

Most errors are due to wrongly generated DCS trees (e.g. wrongly assigned semantic roles) or unimplemented quantifier triggers (e.g. "*neither*") or generalized quantifiers (e.g. "*at least a few*"). These could be addressed by future work.

4.3 Experiments on PASCAL RTE datasets

On PASCAL RTE datasets, strict logical inference is known to have very low recall (Bos and Markert, 2005), so on-the-fly knowledge is crucial in this setting. We test the effect of on-the-fly knowledge on RTE2, RTE3, RTE4 and RTE5 datasets, and compare our system with other approaches.

4.3.1 Impact of on-the-fly knowledge

Results on test data are shown in Table 5. When only primary knowledge is used in inference (the first row), recalls are actually very low; After we activate the on-the-fly knowledge, recalls jump to over 50%, with a moderate fall of precision. As a result, accuracies significantly increase.

[12] http://wordnet.princeton.edu/

[13] http://code.google.com/p/word2vec/

[14] http://metaoptimize.com/projects/wordreprs/

	RTE2			RTE3			RTE4			RTE5		
	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.
Primary	**70.9**	9.8	52.9	**73.2**	7.3	51.1	**89.7**	5.2	52.3	**82.6**	6.3	52.5
+On-the-fly	57.6	**66.5**	**58.8**	63.7	**64.6**	**63.0**	60.0	**57.4**	**59.6**	69.9	**55.7**	**65.8**

Table 5: Impact of on-the-fly knowledge

	RTE2	RTE3	RTE4	RTE5
Bos06	60.6	-	-	-
MacCartney08	-	59.4	-	-
Clark08	-	-	56.5	-
Wang10	**63.0**	61.1	-	-
Stern11	61.6	**67.1**	-	63.5
Stern12	-	-	-	64.0
Our Sys.	58.8	63.0	**59.6**	**65.8**

Table 6: Comparison with other systems

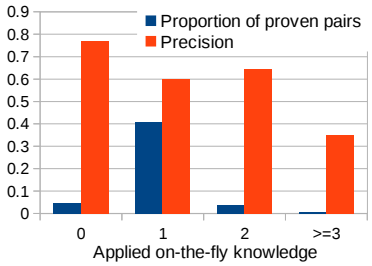

Figure 7: Proportion of proven pairs and their precision, w.r.t. pieces of on-the-fly knowledge.

4.3.2 Comparison to other RTE systems

A comparison between our system and other RTE systems is shown in Table 6. Bos06 (Bos and Markert, 2006) is a hybrid system combining deep features from a theorem prover and a model builder, together with shallow features such as lexical overlap and text length. MacCartney08 (MacCartney and Manning, 2008) uses natural logic to calculate inference relations between two superficially aligned sentences. Clark08 (Clark and Harrison, 2008) is a logic-based system utilizing various resources including WordNet and DIRT paraphrases (Lin and Pantel, 2001), and is tolerant to partially unproven **H** sentences in some degree. All of the three systems pursue a logical approach, while combining various techniques to achieve robustness. The result shows that our system has comparable performance. On the other hand, Wang10 (Wang and Manning, 2010) learns a tree-edit model from training data, and captures entailment relation by tree edit distance. Stern11 (Stern and Dagan, 2011) and Stern12 (Stern et al., 2012) extend this framework to utilize entailment rules as tree transformations. These are more tailored systems using machine learning with many handcrafted features. Still, our unsupervised system outperforms the state-of-the-art on RTE5 dataset.

4.3.3 Analysis

Summing up test data from RTE2 to RTE5, Figure 7 shows the proportion of all proven pairs and their precision. Less than 5% pairs can be proven primarily, with a precision of 77%. Over 40% pairs can be proven by one piece of on-the-fly knowledge, yet pairs do exist in which more than 2 pieces are necessary. The precisions of 1 and 2 pieces on-the-fly knowledge application are over 60%, which is fairly high, given our rough estimation of the similarity score. As a comparison, Dinu and Wang (2009) studied the proportion of proven pairs and precision by applying DIRT rules to tree skeletons in RTE2 and RTE3 data. The proportion is 8% with precision 65% on RTE2, and proportion 6% with precision 72% on RTE3. Applied by our logical system, the noisy on-the-fly knowledge can achieve a precision comparable to higher quality resources such as DIRT.

A major type of error is caused by the ignorance of semantic roles in calculation of similarity scores. For example, though "*Italy beats Kazakhstan*" is not primarily proven from "*Italy is defeated by Kazakhstan*", our system does produce the path alignment "SUBJ(**beat**)OBJ $\approx$ OBJ(**defeat**)SUBJ" with a high similarity score. The impact of such errors depends on the data making methodology, though. It lowers precisions in RTE2 and RTE3 data, particularly in "IE" subtask (where precisions drop under 0.5). On the other hand, it occurs less often in "IR" subtask.

Finally, to see if we "get lucky" on RTE5 data in the choice of word vectors and thresholds, we change the thresholds from 0.1 to 0.7 and draw the precision-recall curve, using two types of word vectors, *Mikolov13* and *Turian10*. As shown in Figure 8, though the precision drops for *Turian10*, both curves show the pattern that our system keeps gaining recall while maintaining precision to a certain level. Not too much "magic" in *Mikolov13* actually: for over 80% pairs, every node in DCS tree of **H** can be covered by a path of length ≤ 5 that

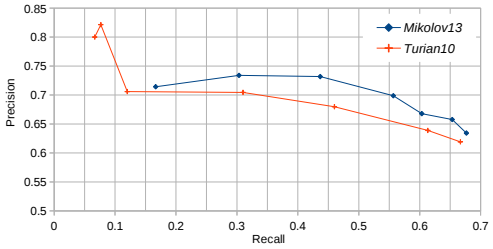

Figure 8: Precision-Recall curve.

has a corresponding path of length ≤ 5 in $\mathbf{T}$ with a similarity score > 0.4.

5 Conclusion and Discussion

We have presented a method of deriving abstract denotation from DCS trees, which enables logical inference on DCS, and we developed a textual inference system based on the framework. Experimental results have shown the power of the representation that allows both strict inference as on FraCaS data and robust reasoning as on RTE data.

Exploration of an appropriate meaning representation for querying and reasoning on knowledge bases has a long history. Description logic, being less expressive than FOL but featuring more efficient reasoning, is used as a theory base for Semantic Web (W3C, 2012). Ideas similar to our framework, including the use of sets in a representation that benefits efficient reasoning, are also found in description logic and knowledge representation community (Baader et al., 2003; Sowa, 2000; Sukkarieh, 2003). To our knowledge, however, their applications to logical inference beyond the use for database querying have not been much explored in the context of NLP.

The pursue of a logic more suitable for natural language inference is not new. For instance, MacCartney and Manning (2008) has implemented a model of natural logic (Lakoff, 1970). While being computationally efficient, various inference patterns are out of the scope of their system.

Much work has been done in mapping natural language into database queries (Cai and Yates, 2013; Kwiatkowski et al., 2013; Poon, 2013). Among these, the (λ-)DCS (Liang et al., 2011; Berant et al., 2013) framework defines algorithms that transparently map a labeled tree to a database querying procedure. Essentially, this is because DCS trees restrict the querying process to a very limited subset of possible operations. Our main contribution, the abstract denotation of DCS trees, can thus be considered as an attempt to characterize a fragment of FOL that is suited for both natural language inference *and* transparent syntax-semantics mapping, through the choice of operations and relations on sets.

We have demonstrated the utility of logical inference on DCS through the RTE task. A wide variety of strategies tackling the RTE task have been investigated (Androutsopoulos and Malakasiotis, 2010), including the comparison of surface strings (Jijkoun and De Rijke, 2005), syntactic and semantic structures (Haghighi et al., 2005; Snow et al., 2006; Zanzotto et al., 2009; Burchardt et al., 2009; Heilman and Smith, 2010; Wang and Manning, 2010), semantic vectors (Erk and Padó, 2009) and logical representations (Bos and Markert, 2005; Raina et al., 2005; Tatu and Moldovan, 2005). Acquisition of basic knowledge for RTE is also a huge stream of research (Lin and Pantel, 2001; Shinyama et al., 2002; Sudo et al., 2003; Szpektor et al., 2004; Fujita et al., 2012; Weisman et al., 2012; Yan et al., 2013). These previous works include various techniques for acquiring and incorporating different kinds of linguistic and world knowledge, and further fight against the knowledge bottleneck problem, e.g. by back-off to shallower representations.

Logic-based RTE systems employ various approaches to bridge knowledge gaps. Bos and Markert (2005) proposes features from a model builder; Raina et al. (2005) proposes an abduction process; Tatu and Moldovan (2006) shows handcrafted rules could drastically improve the performance of a logic-based RTE system.

As such, our current RTE system is at a proof-of-concept stage, in that many of the above techniques are yet to be implemented. Nonetheless, we would like to emphasize that it already shows performance competitive to state-of-the-art systems on one data set (RTE5). Other directions of our future work include further exploitation of the new semantic representation. For example, since abstract denotations are readily suited for data querying, they can be used to verify newly generated assumptions by fact search in a database. This may open a way towards a hybrid approach to RTE wherein logical inference is intermingled with large scale database querying.

Acknowledgments This research was supported by the Todai Robot Project at National Institute of Informatics.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *J. Artif. Int. Res.*, 38(1).

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA.

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

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets markov: Deep semantics with probabilistic logical form. In *Second Joint Conference on Lexical and Computational Semantics (*SEM)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP 2013*.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of EMNLP 2005*.

Johan Bos and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesnt). In *Proceedings of the 2nd PASCAL RTE Challenge Workshop*.

Aljoscha Burchardt, Marco Pennacchiotti, Stefan Thater, and Manfred Pinkal. 2009. Assessing the impact of frame semantics on textual entailment. *Nat. Lang. Eng.*, 15(4).

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL 2013*.

Peter Clark and Phil Harrison. 2008. Recognizing textual entailment with logical inference. In *Proceedings of 2008 Text Analysis Conference (TAC'08)*.

Peter Clark and Phil Harrison. 2009. Large-scale extraction and use of knowledge from text. In *Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP'09)*.

E. F. Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM*, 13(6).

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and et al. 1996. Using the framework. *FraCaS Deliverable D*, 16.

Ido Dagan, O. Glickman, and B. Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*.

Georgiana Dinu and Rui Wang. 2009. Inference rules and their application to recognizing textual entailment. In *Proceedings of EACL 2009*.

Katrin Erk and Sebastian Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*.

Atsushi Fujita, Pierre Isabelle, and Roland Kuhn. 2012. Enlarging paraphrase collections through generalization and instantiation. In *Proceedings of EMNLP 2012*.

Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP 2005*.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL 2010*.

Valentin Jijkoun and Maarten De Rijke. 2005. Recognizing textual entailment: Is word similarity enough? In *Machine Learning Challenge Workshop, volume 3944 of LNCS, Springer.*

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of EMNLP 2013*.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1-2).

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of ACL*, 1.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL 2011*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Nat. Lang. Eng.*, 7(4).

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of Coling 2008*.

Oren Melamud, Jonathan Berant, Ido Dagan, Jacob Goldberger, and Idan Szpektor. 2013. A two level model for context sensitive inference rules. In *Proceedings of ACL 2013*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL 2013*.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Proceedings of NAACL 2007*.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of ACL 2013*.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of EMNLP 2010*.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI 2005*.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT 2002*.

Rion Snow, Lucy Vanderwende, and Arul Menezes. 2006. Effectively using syntax for recognizing false entailment. In *Proceedings of NAACL 2006*.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL 2013*.

John F. Sowa. 2000. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA.

Asher Stern and Ido Dagan. 2011. A confidence model for syntactically-motivated entailment proofs. In *Proceedings of RANLP 2011*.

Asher Stern, Roni Stern, Ido Dagan, and Ariel Felner. 2012. Efficient search for transformation-based inference. In *Proceedings of ACL 2012*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of ACL 2003*.

JanaZ. Sukkarieh. 2003. An expressive efficient representation: Bridging a gap between nlp and kr. In Vasile Palade, RobertJ. Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*. Springer Berlin Heidelberg.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP 2004*.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of ACL 2007*.

Marta Tatu and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proceedings of EMNLP 2005*.

Marta Tatu and Dan Moldovan. 2006. A logic-based semantic approach to recognizing textual entailment. In *Proceedings of the COLING/ACL 2006*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL 2010*.

W3C. 2012. Owl 2 web ontology language document overview (second edition). www.w3.org/TR/owl2-overview/.

Mengqiu Wang and Christopher Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of Coling 2010*.

Hila Weisman, Jonathan Berant, Idan Szpektor, and Ido Dagan. 2012. Learning verb inference rules from linguistically-motivated evidence. In *Proceedings of EMNLP 2012*.

Yulan Yan, Chikara Hashimoto, Kentaro Torisawa, Takao Kawai, Jun'ichi Kazama, and Stijn De Saeger. 2013. Minimally supervised method for multilingual paraphrase extraction from definition sentences on the web. In *Proceedings of NAACL 2013*.

Fabio massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Nat. Lang. Eng.*, 15(4).