Vol.2013-NL-214 No.20
2013/11/15

TIFMO: An Inference-based Textual Entailment Recognition System

Ran Tian[1,a)] Yusuke Miyao[1,b)]

Abstract: The core of TIFMO is an inference engine that operates on *algebraic forms*, in contrast to *logical forms* that are traditionally used to represent the semantics of sentences. Algebraic forms are compositionally constructed *sets* using relational algebra operators, and the meaning of sentences are represented by relations between the sets. Features of the system include: (i) dynamically generated *may-be-missing knowledge* as local alignments between a **T**-**H** pair; and (ii) an evaluation of *logical overlap* that indicates how close we are about to achieve a complete proof. Experimental results and examples are shown to illustrate the usability of the system.

1. Introduction

Recognizing textual entailment (RTE) is the task of determining whether a given textual statement **H** can be inferred by a text passage **T**. One ingenuous approach to this task is to represent **T** and **H** in logical forms, focusing on deriving a proof by which one can infer **H** from **T**. The main problem of this approach is the lack of background knowledge, in that a strict proof of **H** almost always requires some extra information about language or real world, which is not explicitly represented in **T**.

Attempts have been made to remedy this situation through various techniques, including model-building [1], abductive theorem proving [7], and addition of semantic axioms [8]. The model-building provides features related to the cost of a proof (or disproof), whereas abductive theorem proving requires an assumption cost model, thus both of them need to learn on RTE datasets to assign appropriate weights for the costs. These systems do not necessarily outperform shallow features [2] such as word overlap [4], as cost models linguistic and world knowledge, for which an RTE dataset seems too sparse to learn from [9]. On the other hand, addition of semantic axioms requiring manually defined rules could be expensive.

The main contribution of TIFMO is a novel semantic representation framework based on *algebraic forms* (Section 2), which is desirable for two reasons: (i) it provides heuristics for generating *may-be-missing inference rules*, which can be used in a logic-based RTE system to drastically improve the possibility of a complete proof; and (ii) we can use algebraic forms to define *logical overlap*, which enables the evaluation of an inference process even if a complete proof is not available.

Evaluation is conducted on PASCAL RTE and NTCIR RITE datasets (Section 5). We give examples both in English and in Japanese to show how the system works.

2. Algebraic Forms and Logical Overlap

A first order predicate logic (FOL) representation of meaning focuses on relations among *instances*, as symbolized in the formula $\exists c, \exists m, cat(c) \wedge mouse(m) \wedge eat(c, m)$, where c and m are specific instances of cats and mice. However, daily life reasoning seems to prefer relations among "general concepts" or **sets of instances**, as given the sentence "I'm annoyed by mice" we may conclude that "I need a cat", having the knowledge "cats eat mice" rather than "a (specific) cat eats a mouse". Since the knowledge is about general cats and mice, a formalization such as $\forall m, mouse(m) \rightarrow (\exists c, cat(c) \wedge eat(c, m))$ should be untrue.

The framework of *algebraic forms* originates from relational algebra. It compositionally constructs **sets** of instances using relational algebra operators. By considering sets of instances and regarding meanings of sentences as relations between the sets, we acquire the convenience of dynamically generating new relations among general concepts, as in the following example

T *I'm annoyed by mice. Cats eat mice.*
H *I need a cat.*

we may guess relations such as "mice that annoy me" $\subset$ (subset of) "mice eaten by cats" and "cats that eat mice that annoy me" $\subset$ "cats that I need". With these relations, we actually can prove **H** from **T**. (Details on how to generate these relations are described in §4)

2.1 Algebraic Forms

Formally, we fix a "world set" W, and to each content word (or "predicate" in first order predicate logic) w, we assign a symbol I_w representing the set of instances that can be denoted by w. Let $r_1, \ldots, r_n$ be the possible semantic roles that w can take. We then consider I_w to be a subset of $W_{r_1} \times \ldots \times W_{r_n}$. Here, $W_{r_1}, \ldots, W_{r_n}$ are copies of W, and $\times$ denotes the Cartesian product.[*1]

[1] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[a)] tianran@nii.ac.jp
[b)] yusuke@nii.ac.jp

[*1] The order of the Cartesian product is ignored; for example $W_{r_1} \times \ldots \times W_{r_n}$

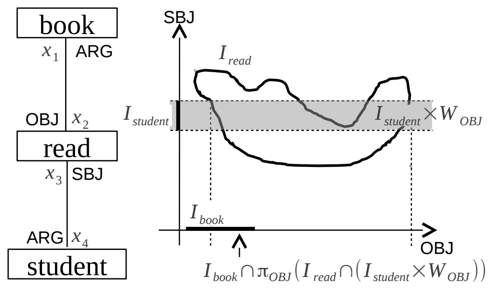

book
x_1 ARG
OBJ x_2
read
x_3 SBJ
ARG x_4
student
SBJ
I_{read}
$I_{student}$
$I_{student} \times W_{OBJ}$
I_{book}
OBJ
$I_{book} \cap \pi_{OBJ}(I_{read} \cap (I_{student} \times W_{OBJ}))$

Fig. 1 Illustration of "books read by students". The corresponding DCS tree is shown on the left.

As an example, since the word "student" usually appears as an argument in a predicate-argument structure (e.g. $read(student, book)$), we consider the word to have semantic role `ARG`, and regard $I_{student}$ as a subset of W_{ARG}. On the other hand, the word "read" has semantic roles `SBJ` and `OBJ`, so $I_{read} \subset W_{SBJ} \times W_{OBJ}$.

Since W_{SBJ} and W_{ARG} are copies to each other, we can also regard $I_{student}$ as a subset of W_{SBJ}, which is denoted by $\iota_{SBJ}(I_{student})$. Then the composition "students read" can be represented by the set $I_{read} \cap (\iota_{SBJ}(I_{student}) \times W_{OBJ})$, as illustrated in Figure 1: the 1-dimensional set $\iota_{SBJ}(I_{student})$ is drawn on the `SBJ`-axis, and 2-dimensional set I_{read} shown as a region on the `SBJ`-`OBJ` plane, so $I_{read} \cap (\iota_{SBJ}(I_{student}) \times W_{OBJ})$ is the intersection of the shadow with this region.

This composition process is very similar to a relational database: we can imagine a table "Read" of two fields, `SBJ` and `OBJ` (Table 1), and a table "Students" containing names such as $\{Mary, \ldots\}$. Then "students read" should be those records in the "Read" table, the `SBJ` value of which is the name of a student. Furthermore, the concept "books read by students" should be intersection of the values in the table "Book" and the `OBJ` values in "students read", which is also illustrated in Figure 1.

The set operators introduced herein are:

- $\times$: The Cartesian product.
- $\cap$: The intersection of two sets.
- π: Projection. We denote the i-th projection of $W_{r_1} \times \ldots \times W_{r_n} \to W$ by π_{r_i}.
- ι: Relabeling. The identification map $W \to W_r$ is denoted by ι_r. Thus, for any $A \subset W$, we consider $\iota_r(A)$ to be a subset of W_r.
- Division operators, which are operators that resemble the division operator in relational algebra. Details are described in §3.2.
- Selection operators, which produce a specific subset of A for any $A \subset W$. Details are described in §3.3.

2.2 Logical Overlap

We consider three fundamental relations of sets: the non-emptiness ($\neq \emptyset$), subsumption ($\subset$) and disjointness ($\|$). The meaning of a sentence is represented by relations between algebraic forms, as shown in the following examples:

and $W_{r_n} \times \ldots \times W_{r_1}$ are thought to be the same. This does not cause confusion because every component of the product is uniquely labeled by a semantic role.

Table 1 "Read"

`SBJ`	`OBJ`
Mark	New York Times
Mary	A Tale of Two Cities
...	...

- *Cats eat mice*: $I_{eat} \cap (I_{cat} \times I_{mouse}) \neq \emptyset$.
- *All men are mortal*: $I_{man} \subset I_{mortal}$.
- *No animals are harmed*: $I_{animal} \parallel \pi_{OBJ}(I_{harm})$.

For a **T**-**H** pair, let the meaning of **H** be represented by a set of relations, $\mathcal{R}_0$, among a set of algebraic forms, $\mathcal{A}_0$. Simpler algebraic forms can be derived from elements in $\mathcal{A}_0$, for example we can consider sub-trees of the expression tree of each algebraic form. Now we fix a set $\mathcal{A}_1$ derived from $\mathcal{A}_0$, then the logical overlap is defined as the rate $r = |\mathcal{R}'_1|/|\mathcal{R}_1|$, where $\mathcal{R}_1$ is the set of relations among $\mathcal{A}_1$ that can be infered from **H**, and $\mathcal{R}'_1$ is the subset of $\mathcal{R}_1$ that can be infered from **T**. The norm $|\cdot|$ is either the cardinality of a set, or a weighted sum of its elements, with some reasonable weight function such as the expression length of algebraic forms that appear in a relation.

As an example, consider the following **T**-**H** pair:

T *John reads a book.*
H *A boy reads a book.*

We have $\mathcal{A}_0 = \{I_{read} \cap (I_{boy} \times I_{book})\}$ and $\mathcal{R}_0 = \{I_{read} \cap (I_{boy} \times I_{book}) \neq \emptyset\}$. For $\mathcal{A}_1$ we may put $\mathcal{A}_1 = \{I_{read}, I_{boy}, I_{book}, I_{read} \cap (I_{boy} \times I_{book}), I_{read} \cap (W_{SBJ} \times I_{book}), I_{read} \cap (I_{boy} \times W_{OBJ})\}$. Then $\mathcal{R}_1$ is the set of relations saying all elements in $\mathcal{A}_1$ are non-empty. Now if we don't have the knowledge "John is a boy", the subset of relations provable from **T** should be $\mathcal{R}'_1 = \{I_{read} \cap (W_{SBJ} \times I_{book}) \neq \emptyset, I_{read} \neq \emptyset, I_{book} \neq \emptyset\}$.

Logical overlap can be viewed as an extension of word overlap, in that if we put $\mathcal{A}_1 = \{I_w \mid w \text{ is a content word of } \mathbf{H}\}$, then in many cases we have $\mathcal{R}_1 = \{I_w \neq \emptyset \mid w \text{ is a content word of } \mathbf{H}\}$, and the cardinality of $\mathcal{R}'_1$ will be the number of content words of **H** that also appear in **T**, as shown in the above example. On the other hand, if we let $\mathcal{A}_1$ contain all elements in $\mathcal{A}_0$, then a logical overlap equal to 1 means an entailment relation exactly holds, since all relations shaping the meaning of **H** are proven from **T**. In order to make the value of a logical overlap rate more reasonable, we may assign a larger weight on the relation $I_{read} \cap (W_{SBJ} \times I_{book}) \neq \emptyset$ than $I_{book} \neq \emptyset$, since the former is harder to prove.

The derivation of $\mathcal{A}_1$ from $\mathcal{A}_0$, and the examination of $\mathcal{R}_1$ instead of $\mathcal{R}_0$, is a strategy parallel to the one in an FOL setting that, besides the single proposition corresponding to **H**, trying to prove some "partial statements" (propositions that should be true if **H** is true) as well. The difference is that, algebraic forms suggest a systematic method to generate partial statements that are intuitively meaningful, in the sense that all relations in $\mathcal{R}_1$ are directly related to the original sentence **H**, such as $I_{read} \cap (W_{SBJ} \times I_{book}) \neq \emptyset$ corresponding to "someone reads a book". This may not be true in an FOL setting, for example $(\exists x, boy(x)) \to (\exists y, book(y))$ is a true statement if **H** is true, but the meaning seems quite irrelevant.

3. DCS Trees

For the conversion of natural language sentences to algebraic forms, we adopt the DCS framework proposed in [6], which converts natural language sentences to semantically annotated dependency trees (DCS-trees) and then to database queries. Since the logic behind a relational database is relational algebra, the database queries can be rewritten as algebraic formulas using relational algebra operators, i.e. algebraic forms. However, the original DCS framework was developed for the purpose of building a natural language interface for some existing databases, and the semantically annotated dependency trees are latently learned from instances in the databases and QA datasets. For our purpose, it is unrealistic to assume the existence of relational databases for learning for open-domain semantic processing and real-world RTE tasks. Thus, we use off-the-shelf dependency parsers and apply rule-based semantic annotations to convert dependency trees to DCS-trees, making use of the fact that these two trees are fairly similar. Therefore, the semantic annotations we add to dependency trees are different from those in the original DCS, although the basic version remains the same. As such, we also refer to the annotated trees as “DCS trees”.

3.1 Basic Version

A basic version DCS tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ is defined as follows: $\mathcal{T}$ is a rooted tree in which 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'), which can be taken by the word $w(\sigma)$ and $w(\sigma')$, respectively (Figure 1). For any edge (σ, σ') labeled with roles (r, r'), we refer to the pairs (σ, r) and (σ', r') as *germs*.

DCS trees are related to database queries or algebraic forms, through the idea that they can define constraint satisfaction problems (CSP). For a DCS tree of basic version, each germ (σ, r) corresponds to a variable $x_{\sigma_r} \in W$, and the constraints are:

(1) If an edge (σ, σ') is labeled with roles (r, r'), then $x_{\sigma_r} = x_{\sigma'_{r'}}$.
(2) For a node σ, let $(\sigma, r_1), \ldots, (\sigma, r_k)$ be germs, and $I_{w(\sigma)} \subset W_{r_1} \times \ldots \times W_{r_k}$. Then $(\iota_{r_1}(x_{\sigma_{r_1}}), \ldots, \iota_{r_k}(x_{\sigma_{r_k}})) \in I_{w(\sigma)}$.

An instance $\alpha \in W$ is said to be *consistent* to a germ (σ, r), if there exists a solution to the CSP such that $x_{\sigma_r} = \alpha$. The set of consistent instances to the germ (σ, r) is refered to as the *denotation* of (σ, r), denoted by $D(\sigma, r; \mathcal{T})$.

For example, the DCS tree for the phrase “books read by students” (Figure 1) defines a CSP on four variables, x_1, x_2, x_3 and x_4, corresponding to the four germs, $(book, ARG)$, $(read, OBJ)$, $(read, SBJ)$ and $(student, ARG)$, respectively. Constraints imposed by the CSP are as follows:

(1) $x_1 = x_2$, $x_3 = x_4$.
(2) $x_1 \in I_{book}$, $(\iota_{OBJ}(x_2), \iota_{SBJ}(x_3)) \in I_{read}$, $x_4 \in I_{student}$.

So the denotation of germ $(book, ARG)$ is the set $\{x \in W \mid x \in I_{book} \wedge (\exists y, y \in I_{student} \wedge (\iota_{OBJ}(x), \iota_{SBJ}(y)) \in I_{read})\}$, representing the concept “books read by students”.

Point of the DCS framework is that the CSP defined by a DCS tree can be explicitly solved, in the sense that the denotation of each germ can be recursively calculated as algebraic forms (cf. §2.1 in [6]). In the above example, the denotation of $(book, ARG)$ can also be written by $I_{book} \cap \pi_{OBJ}(I_{read} \cap (\iota_{SBJ}(I_{student}) \times W_{OBJ}))$.

3.2 Universal Quantifiers

For most declarative sentences, a large part of their syntactic trees can be converted into basic version DCS trees, and the meaning, at least approximately, can be formalized by the satisfiability of the corresponding CSP (e.g. Figure 3). Note that the satisfiability is logically equivalent to the non-emptiness of a denotation of any germ of the DCS tree, so in most cases the meaning of a sentence can be represented by the non-emptiness of some algebraic forms. However, sometimes we need to state properties held by all elements in a denotation, which is parallel to a universal quantifier in an FOL language, in contrast to the existential quantifier corresponding to non-emptiness. Two relations and two division operators serve for this purpose:

- $\subset$: Subsumption. For two denotations D_1 and D_2, the relation $D_1 \subset D_2$ represents the statement $\forall x, x \in D_1 \to x \in D_2$.
- $\|$: Disjointness. $D_1 \parallel D_2$ corresponds to the statement $\forall x, x \in D_1 \to x \notin D_2$.
- $\mathfrak{q}_{all}$: The “division” operator in relational algebra, for a set $A \subset W_{r_1} \times W_{r_2} \times \ldots W_{r_n}$, the set $\mathfrak{q}_{all}(A, \iota_{r_1}(D))$ is the largest set $X \subset W_{r_2} \times \ldots \times W_{r_n}$ such that $\iota_{r_1}(D) \times X \subset A$. For example, the denotation of “chocolate” in the phrase “chocolates that all kids like” can be represented by $I_{chocolate} \cap \mathfrak{q}_{all}(I_{like}, \iota_{SBJ}(I_{kid}))$ (Figure 2).
- $\mathfrak{q}_{no}$: Similar to $\mathfrak{q}_{all}$, whereas $\mathfrak{q}_{no}(A, \iota_{r_1}(D))$ is the largest set X such that $\iota_{r_1}(D) \times X \parallel A$.

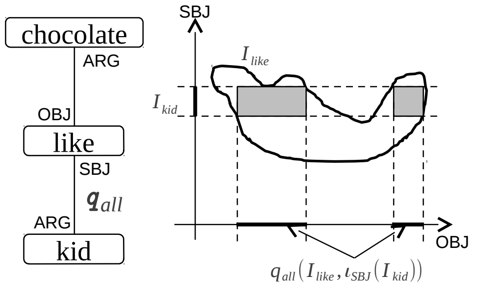


Fig. 2 Division Operator

In practice, universal quantifiers are neither dense in usual sentences, nor critical in most **T**-**H** pairs. However, universal quantifiers are useful for stating common sense knowledge, for example we can represent hypernyms $I_{cat} \subset I_{animal}$, mutually exclusive events $I_{rise} \parallel I_{fall}$, derivations $I_{criminal} \subset \pi_{SBJ}(I_{commit}, \iota_{OBJ}(I_{crime}))$ (all criminals commit a crime), and world knowledge $I_{champion} \subset \mathfrak{q}_{all}(I_{win}, \iota_{OBJ}(I_{game}))$ (champions win all the games).

3.3 Selection

Selection operators select subsets of specific properties out of concepts. Examples are: (i) superlatives, $\mathfrak{s}_{highest}(\pi_{ARG}(I_{mountain} \cap (W \times \iota_{LOC}(I_{Asia}))))$ (the highest mountain in Asia); and (ii) numerics, $\mathfrak{s}_{two}(I_{soldier} \cap I_{wounded})$ (two wounded soldiers).

We implement selection operators as markers assigned to denotations, applying special axioms designed for each type of selection. Currently for superlatives we have downward monotonicity: $A \subset B$ & $\mathfrak{s}_{highest}(B) \subset A \Rightarrow \mathfrak{s}_{highest}(B) \subset \mathfrak{s}_{highest}(A)$, and for numerics we have awareness of size: $\mathfrak{s}_a(A) \subset \mathfrak{s}_b(B)$ & $a > b \Rightarrow \perp$. New rules can be added if necessary.

3.4 Negation

Currently we implement two types of negations: (i) atomic negation, for each content word w we allow negation $\bar{w}$ of that word, characterized by the property $I_w \parallel I_{\bar{w}}$; and (ii) root negation, for sentences having a meaning that can be represented by the non-emptiness of an algebraic form α, we can negate the entire sentence by claiming that $\alpha \parallel \alpha$.

4. The TIFMO System

As an RTE system, TIFMO operates as follows:

(1) For a **T**-**H** pair, it applies preprocessing, parsing, and coreference resolution on the natural language sentences.
(2) Rule-based semantic annotations are performed to convert dependency parses into DCS trees. The meaning of sentences are translated into relations between algebraic forms.
(3) Add the premise **T** and solid knowledge to the inference engine, and try to prove **H**.
(4) If **H** is not proven, it compares DCS trees of **T** with DCS trees of **H**, in order to dynamically generate *may-be-missing rules*, in the form of aligned paths in DCS trees.
(5) Use some similarity measure to estimate the confidence of generated paths alignments. Then select the most confident one as a new piece of knowledge.
(6) Convert the selected path alignments into new relations between algebraic forms, add the relations to the inference engine, and try to prove **H** again.
(7) Record the selected alignments, their confidence, and the gain of logical overlap rates after applying the corresponding knowledge. If **H** is not proven yet, go back to Step 4. If **H** is proven or no more knowledge can be generated, break the loop.
(8) The records of selected alignments, confidence, and the gain of logical overlaps are used by a classifier to output the final label.

Details of each step are described in subsections.

4.1 Parsing and Semantic Annotation

For English, we use BIU Number Normalizer[*2] for preprocessing, and Stanford CoreNLP[*3] for parsing and coreference resolution. The collapsed dependency tree output by the Stanford parser is designed to maximize dependencies between content words[*4], therefore suitable for our purpose, and the Stanford dependency labels provide rich information for semantic annotation. Semantic role labels are produced by pattern matching according to Stanford dependency labels and POS tags. Determiners such as "all", "every", "each" and "no" trigger universal quantifiers. Numeric expressions and superlatives are recognized as selections.

For Japanese, we use normalizeNumexp[*5] for preprocessing, and we use Cabocha [5] and Syncha [3] for dependency parsing and zero anaphora resolution. Since the dependencies output by Cabocha are between Japanese chunks, which can have compound words in the content word part, we use NihongoGoiTaikei[*6] and Wikipedia page titles as dictionaries to apply max-length matching from the left to separate compound words. Rules based on function words (categorized by Tsutusji Function Words Dictionary[*7]) and verb frames in NihongoGoiTaikei are used to obtain semantic roles.

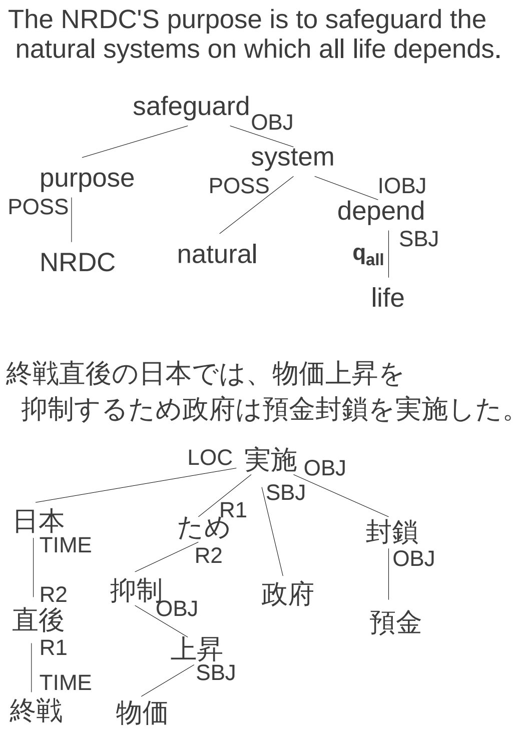


Fig. 3 DCS trees. Unmarked germs are labeled with semantic role ARG.

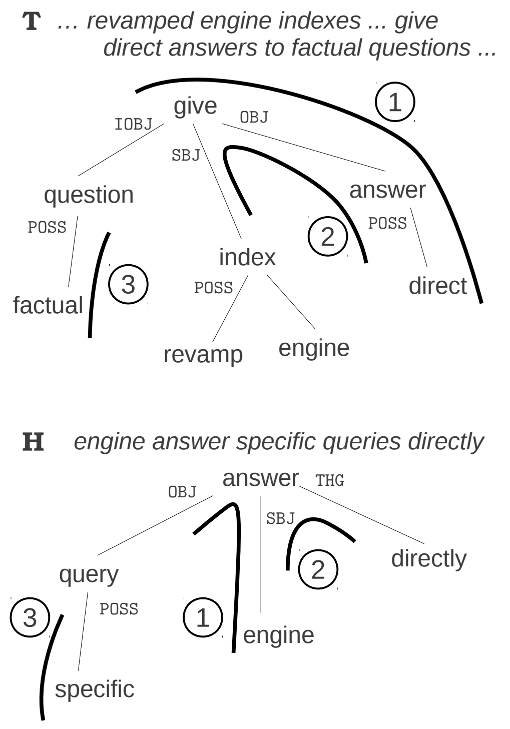
The semantic roles we use (for both Japanese and English) are: `ARG`, `SBJ`, `OBJ`, `IOBJ`, `POSS`, `TIME`, `LOC`, `R1` and `R2`. Role `POSS` is used for possessions, `R1` and `R2` are used in directed binary relations. Finally obtained DCS trees are mixed structures of dependencies, predicate-argument structure, semantic roles, quantifiers and relationships (e.g. Figure 3).

4.2 Knowledge Resources

For English, synonyms, hypernyms, antonyms and derivations from WordNet[*8] are regarded as solid knowledge and applied at the beginning of a proof. We wrote several templates for derivations. As an example, if a verb v and a noun n are derivationally related, we have templates such as $\pi_{ARG}(I_n \cap W_{SBJ} \times \iota_{POSS}(X)) = \pi_{SBJ}(I_v \cap W_{SBJ} \times \iota_{OBJ}(X))$, representing knowledge such as "X's director = the subject who direct X". We also adopt a stopword list used for MySQL full text search[*9], and consider I_w of stopword w to be Cartesian product of W.

For Japanese, synonym relations are extracted from BunruiGoiHyo[*10] and Wikipedia redirect[*11], hypernyms are obtained from ALAGIN verb entailment database[*12] and Wikipedia hyponymy extraction tool[*13], and antonyms are extracted from Ni-

[*2] `u.cs.biu.ac.il/~nlp/downloads/normalizer.html`
[*3] `nlp.stanford.edu/software/corenlp.shtml`
[*4] `nlp.stanford.edu/software/dependencies_manual.pdf`
[*5] `www.cl.ecei.tohoku.ac.jp/~katsuma/software/normalizeNumexp/`
[*6] `www.kecl.ntt.co.jp/icl/lirg/resources/GoiTaikei/`
[*7] `kotoba.nuee.nagoya-u.ac.jp/tsutsuji/`
[*8] `wordnet.princeton.edu/wordnet/`
[*9] `dev.mysql.com/doc/refman/5.1/en/fulltext-stopwords.html`
[*10] `www.ninjal.ac.jp/archives/goihyo/`
[*11] `code.google.com/p/wikipedia-redirect/`
[*12] `alaginrc.nict.go.jp/resources/nictmastar/li-resource-info/li-resource-outline.html#A-2`
[*13] `alaginrc.nict.go.jp/hyponymy/index.html`



① "give direct answer (to)" – "engine answer (OBJ)"
② "give (direct) answer" – "answer (directly)"
③ "factual (question)" – "specific (query)"

Fig. 4 Dynamically Generated Alignments

hongoGoiTaikei and Kojien dictionary[*14].

4.3 Logical Overlap

The set of denotations of all germs of DCS trees of **H** is denoted by $\mathcal{A}_0$. The set $\mathcal{A}_1$ consists of algebraic forms derived from $\mathcal{A}_0$, which are obtained as follows: for any $\alpha \in \mathcal{A}_0$, consider its expression tree $\mathcal{E}_\alpha$; for any node n in $\mathcal{E}_\alpha$, consider the subtree rooted at n and its complement. For example, from the algebraic form $I_{book} \cap \pi_{OBJ}(I_{read} \cap (\iota_{SBJ}(I_{student}) \times W_{OBJ}))$ representing "books read by students", and the node I_{read} in the expression tree, we can obtain a subtree rooted at I_{read} as $I_{read} \cap (\iota_{SBJ}(I_{student}) \times W_{OBJ})$ ("students read"), and its complement I_{book}. The weight of each algebraic form are set to the number of nodes in its expression tree that correspond to non-stopwords. In the above example, weight of I_{book} is 1, and weight of $I_{read} \cap (\iota_{SBJ}(I_{student}) \times W_{OBJ})$ is 2 because there are two non-stopwords, "student" and "read", in the expression.

4.4 Inference Engine

As we mentioned in §2.2, considering the relations among various algebraic forms is equivalent to considering several partial statements in an FOL setting, and we must determine which of these relations can be proven. This becomes a non-trivial problem when relations increase, if we naively translate algebraic forms to sets in an FOL language and rely on theorem provers. The inference engine of TIFMO directly operates on algebraic forms. The engine regards I_w for each content word w as *constants* rather than *predicates*, relational algebra operators as *functions*, and algebraic forms as *terms*, rather than *sentences*. As a result, relations such as $I_{book} \neq \emptyset$ become *atomic sentences* rather than *first order formulas* such as $\exists x, book(x)$. The inference process for atomic sentences can be treated simply as in propositional logic. The cost is that we should write axioms for relational algebra operators, such as $(A\cap B)\times(C\cap D) = (A\times C)\cap(B\times D)$, by ourselves. Approximately 30 axioms for relational algebra are implemented, which enables most common theorems to be proven very quickly.

4.5 May-be-missing Knowledge

To generate may-be-missing knowledge, we simply compare all paths in DCS trees of **T** and **H**, according to their denotations. For a path p_1 joining two germs (σ_1, r_1) and (σ'_1, r'_1) in a DCS tree $\mathcal{T}_1$ of **T**, and a path p_2 joining (σ_2, r_2) and (σ'_2, r'_2) in $\mathcal{T}_2$ of **H**, we generate an "alignment" of p_1 and p_2, if (i) the denotation $D(\sigma_1, r_1; \mathcal{T}_1)$ and $D(\sigma_2, r_2; \mathcal{T}_2)$ subsume a common algebraic form; and (ii) σ'_1 and σ'_2 are leaf nodes, **or** $D(\sigma'_1, r'_1; \mathcal{T}_1)$ and $D(\sigma'_2, r'_2; \mathcal{T}_2)$ also subsume a common algebraic form. All possible alignments are generated and assigned a confidence score. Alignments of the highest confidence are regarded as knowledge and are added to the inference engine. The inference rule corresponding to an alignment is generated as follows: let $\mathcal{T}_{p_1}$ be the subgraph of $\mathcal{T}_1$ corresponding to the path p_1, and let $\mathcal{T}_{p_2}$ be the subgraph of $\mathcal{T}_2$ corresponding to p_2. Then we generate the relation $D(\sigma_1, r_1; \mathcal{T}_{p_1}) \subset D(\sigma_2, r_2; \mathcal{T}_{p_2})$.

4.6 Confidence Score

Each path alignment should be assigned a confidence score. This is actually the most flexible part of TIFMO system, as we can experiment on various methods of calculating similarities of short phrases.

Currently for English, we have used distributional similarities calculated from Google Web 1T 5-gram corpus[*15]. For a word set S, we calculate its distribution vector as follows: for a 5-gram $w_1, \ldots, w_5$, if $S \subset \{w_1, \ldots, w_5\}$ and $w \in \{w_1, \ldots, w_5\} \setminus S$ is a non-stopword, we add 1 to coordinate w[*16]. Then the similarity of two phrases are obtained as the cosine similarity of their distribution vectors. As for Japanese, we use BunruiGoiHyo categories to calculate a similarities between each two words, and take the average as the similarity of two phrases.

The rationale for combining strict logical inference with rough similarities in this way, is that (i) we only need to calculate similarities for short phrases, and then combine them via compositional semantics, which overcomes the sparsity problem for long sentences; and (ii) though the similarity measure is inaccurate and noisy, we can expect that strict logical inference acts as "filters", to cut off irrelevant alignments and highlight critical ones that are actually necessary for proving **H** from **T**.

As an example, we present the following pair taken from PASCAL RTE2 development set.

T *The revamped engine indexes more pages than before, can give direct answers to factual questions, and features tools*

[*14] www.iwanami.co.jp/kojien/

[*15] www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2006T13

[*16] For searching of 5-grams we used SSGNC (code.google.com/p/ssgnc/).

Table 2 Evaluation on PASCAL RTE

	RTE1			RTE2			RTE3			RTE5		
	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.
Pre	**71.43**	11.25	53.38	**81.43**	14.25	55.50	**74.42**	7.80	51.38	**71.43**	10.00	53.00
Post	57.98	**47.25**	**56.50**	61.64	**56.25**	**60.63**	58.77	**46.59**	**55.88**	68.45	**42.67**	**61.50**

Table 3 Evaluation on NTCIR RITE

	RITE1-Exam-dev			RITE1-Exam-test			RITE2-ExamBC-dev			RITE2-ExamBC-test		
	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.
Pre	50.00	3.92	59.12	60.00	1.66	59.28	66.67	3.81	59.61	**60.00**	1.73	**61.61**
Post	**64.18**	**21.08**	**62.93**	**67.21**	**22.65**	**63.80**	**71.70**	**18.10**	**63.33**	47.62	**11.56**	60.94

Table 4 Ablation Test of Features (English)

SWO	HPF	LOL	RTE1	RTE2	RTE3	RTE5
✓			55.75	57.25	66.25	59.67
	✓		55.88	60.50	57.88	61.67
		✓	58.25	60.88	59.75	60.50
✓	✓		57.38	**63.25**	64.13	63.17
✓		✓	58.00	62.88	**67.13**	60.67
	✓	✓	59.00	61.25	59.88	63.67
✓	✓	✓	**60.00**	62.38	64.63	**65.17**

Table 5 Ablation Test of Features (Japanese)

SWO	HPF	LOL	RITE1-Exam	RITE2-ExamBC
✓			65.84	**64.29**
	✓		64.03	59.15
		✓	59.28	61.16
✓	✓		65.84	60.71
✓		✓	**66.97**	63.62
	✓	✓	64.48	60.94
✓	✓	✓	65.61	61.38

to help people create detailed queries.

H *The engine can answer specific queries directly.*

A part of the DCS tree of **T**, and the DCS tree of **H**, output by TIFMO, are shown in Figure 4. At the begining of proof, WordNet contributes the knowledge $I_{question} = I_{query}$, $I_{direct} = I_{directly}$ and $\pi_{SBJ}(I_{answer/V}) = \pi_{SBJ}(I_{give} \cap W \times \iota_{OBJ}(I_{answer/N}))$. The aligned paths selected by TIFMO are marked as ①, ② and ③. After these rules are generated, **H** is proven from **T**. Though alignment ① is incorrectly selected because of their frequent co-occurence with "questions", we should be noted that **H** can already be infered from **T** by the generated alignments ② and ③, even without ①. It shows that logical inference can actually provide useful information for good alignments.

5. Experiments

We evaluate the system on PASCAL RTE and NTCIR RITE datasets. Firstly we show how many pairs can be completely proven by injection of may-be-missing rules. The result is shown in Table 2 and Table 3. We produce a "Y" label if **H** is completely proven[*17]. The row "Pre" is the result of strict proof before injection of may-be-missing knowledge, and the row "Post" is the result after injection.

From the tables we can see drastic improvement of recall brought by the may-be-missing knowledge, and in most cases drop of precision is moderate, which significantly increases the accuracy. For RITE1-Exam-dev, RITE1-Exam-test and RITE2-ExamBC-dev datasets, the precisions even increase.

Secondly we show the effect of using logical overlap rates to evaluate incomplete proofs. For this purpose, we use logical overlaps as features to train supervised classifiers. Results on PASCAL RTE datasets are shown in Table 4. We use three types of features: `SWO` is a simple word overlap feature, `HPF` is a feature showing if there is a complete proof (after may-be-missing knowledge injection), and `LOL` are weighted sums of logical overlap gains, the weights are obtained by confidence scores of the dynamically generated alignments that are responsible to the gain.

Table 4 shows that a combination of `SWO` and logical features can usually boost the performance, and in particular, `HPF` and `LOL` features are significantly different from `SWO` feature, while their stand-alone performances are comparable.

Results on NTCIR RITE datasets are shown in Table 5. We can see feature `LOL` slightly improves the accuracy in RITE1-Exam. In RITE2-ExamBC, feature `SWO` performs the best, a result compatible to the precision drop appeared in Table 3.

6. Conclusion

We introduced TIFMO, an inference-based system for textual entailment recognition. The behind semantics framework using algebraic forms are described, and two features, (i) dynamical generation of may-be-missing rules and (ii) evaluation of incomplete proving process using logical overlaps, are discussed. The system is evaluated on PASCAL RTE and NTCIR RITE datasets.

References

[1] Bos, J. and Markert, K.: Recognising Textual Entailment with Logical Inference, *Proceedings of EMNLP 2005* (2005).

[2] Bos, J. and Markert, K.: When logical inference helps determining textual entailment (and when it doesnt), *Proceedings of the 2nd PASCAL RTE Challenge Workshop* (2006).

[3] Iida, R. and Poesio, M.: A Cross-Lingual ILP Solution to Zero Anaphora Resolution, *Proceedings of ACL 2011* (2011).

[4] Jijkoun, V. and De Rijke, M.: Recognizing textual entailment: Is word similarity enough?, *Machine Learning Challenge Workshop, volume 3944 of LNCS, Springer.* (2005).

[5] Kudo, T. and Matsumoto, Y.: Japanese Dependency Analyisis using Cascaded Chunking, *Proceedings of CoNLL 2002* (2002).

[6] Liang, P., Jordan, M. and Klein, D.: Learning Dependency-Based Compositional Semantics, *Proceedings of ACL 2011* (2011).

[7] Raina, R., Ng, A. Y. and Manning, C. D.: Robust textual inference via learning and abductive reasoning, *Proceedings of AAAI 2005* (2005).

[8] Tatu, M. and Moldovan, D.: A Semantic Approach to Recognizing Textual Entailment, *Proceedings of EMNLP 2005* (2005).

[9] Zanzotto, F. m., Pennacchiotti, M. and Moschitti, A.: A machine learning approach to textual entailment recognition, *Nat. Lang. Eng.*, Vol. 15, No. 4 (2009).

[*17] In practice, the meaning of some **H** may correspond to more than one statements. So we actually output a "Y" when the proportion of proven statements is > 0.5.