Annotating by Proving using SemAnTE

Assaf Toledo[1] **Stavroula Alexandropoulou**[1] **Sophie Chesney**[2]

Robert Grimm[1] **Pepijn Kokke**[1] **Benno Kruit**[3]

Kyriaki Neophytou[1] **Antony Nguyen**[1] **Yoad Winter**[1]

[1] - Utrecht University [2] - University College London [3] - University of Amsterdam
{a.toledo,s.alexandropoulou,y.winter}@uu.nl
sophie.chesney.10@ucl.ac.uk, {pepijn.kokke,bennokr}@gmail.com
{r.m.grimm,k.neophytou,a.h.nguyen}@students.uu.nl

Abstract

We present SemAnTE, a platform for marking and substantiating a semantic annotation scheme of textual entailment according to a formal model. The platform introduces a novel approach to annotation by providing annotators immediate feedback whether the data they mark are substantiated: for positive entailment pairs, the system uses the annotations to search for a formal logical proof that validates the entailment relation; for negative pairs, the system verifies that a counter-model can be constructed. By integrating a web-based user-interface, a formal lexicon, a lambda-calculus engine and an off-the-shelf theorem prover, this platform facilitates the creation of annotated corpora of textual entailment. A corpus of several hundred annotated entailments is currently in preparation using the platform and will be available for the research community.

1 Introduction

The Recognizing Textual Entailment (RTE) challenges (Dagan et al., 2006) advance the development of systems that automatically determine whether an entailment relation obtains between a naturally occurring text *T* and a manually composed hypothesis *H*. The RTE corpus (Bar Haim et al., 2006; Giampiccolo et al., 2008), which is currently the only available resource of textual entailments, marks entailment candidates as positive/negative.[1] For example:

Example 1

- T: The book contains short stories by the famous Bulgarian writer, Nikolai Haitov.
- H: Nikolai Haitov is a writer.[2]
- Entailment: Positive

This categorization does not indicate the linguistic phenomena that underlie entailment or their contribution to inferential processes. In default of a gold standard identifying linguistic phenomena triggering inferences, entailment systems can be compared based on their performance, but the inferential processes they employ to recognize entailment are not directly accessible and consequently cannot be either evaluated or improved straightforwardly.

We address this problem by elucidating some of the central inferential processes underlying entailments in the RTE corpus, which we model formally within a standard semantic theory. This allows us not only to indicate linguistic phenomena that are involved in the recognition of entailment by speakers, but also to provide formal proofs that substantiate the annotations and explain how the

[1]Pairs of sentences in RTE 1-3 are categorized in two classes: *yes-* or *no-entailment*; pairs in RTE 4-5 are categorized in three classes: *entailment*, *contradiction* and *unknown*. We label the judgments *yes-entailment* from RTE 1-3 and *entailment* from RTE 4-5 as *positive*, and the other judgments as *negative*.

[2]Pair 622 from the development set of RTE 2.

modeled phenomena interact and contribute to the recognition process. In this sense the we adopt an *Annotating by Proving* approach to textual entailment annotation.

The annotation work is done using the SemAnTE (Semantic Annotation of Textual Entailment) platform, which incorporates a web-based user-interface, a formal lexicon, a lambda-calculus engine and an off-the-shelf theorem prover. We are currently using this platform to build a new corpus of several hundred annotated entailments comprising both positive and negative pairs. We decided to focus on the semantic phenomena of appositive, restrictive and intersective modification as these semantic phenomena are prevalent in the RTE datasets and can be annotated with high consistency, and as their various syntactic expressions can be captured by a limited set of concepts.[3] In the future, we plan to extend this sematic model to cover other, more complex phenomena.

2 Semantic Model

To model entailment in natural language, we assume that entailment describes a *preorder* on sentences. Thus, any sentence trivially entails itself (reflexivity); and given two entailments $T_1 \Rightarrow H_1$ and $T_2 \Rightarrow H_2$ where H_1 and T_2 are identical sentences, we assume $T_1 \Rightarrow H_2$ (transitivity). We use a standard model-theoretical extensional semantics, based on the simple *partial order* on the domain of *truth-values*. Each model M assigns sentences a truth-value in the set $\{0, 1\}$. Such a Tarskian theory of entailment is considered adequate if the intuitive entailment preorder on sentences can be described as the pairs of sentences T and H whose truth-values $[\![T]\!]^M$ and $[\![H]\!]^M$ satisfy $[\![T]\!]^M \leq [\![H]\!]^M$ for all models M.

We use annotations to link between textual representations in natural language and model-theoretic representations. This link is established by marking the words and structural configurations in T and H with lexical items that encode semantic meanings for the linguistic phenomena that we model. The lexical items are defined formally in a lexicon, as illustrated in Table 1 for major lexical categories over type:s e for *entities*, t for *truth-values*, and their functional compounds.

[3]This conclusion is based on an analysis of RTE 1-4, in which these modification phenomena were found to occur in 80.65% of the entailments and were annotated with cross-annotator agreement of 68% on average.

Category	Type	Example	Denotation
Proper Name	e	Dan	**dan**
Indef. Article	$(et)(et)$	a	A
Def. Article	$(et)e$	the	ι
Copula	$(et)(et)$	is	IS
Noun	et	book	**book**
Intrans. verb	et	sit	**sit**
Trans. verb	eet	contain	**contain**
Pred. Conj.	$(et)((et)(et))$	and	AND
Res. Adj.	$(et)(et)$	short	$R_m(\textbf{short})$
Exist. Quant.	$(et)(et)t$	some	SOME

Table 1: Lexicon Illustration

Denotations that are assumed to be arbitrary are given in boldface. For example, the intransitive verb *sit* is assigned the type et, which describes functions from entities to a truth-values, and its denotation **sit** is an arbitrary function of this type. By contrast, other lexical items have their denotations restricted by the given model M. As illustrated in Figure 1, the coordinator *and* is assigned the type $(et)((et)(et))$ and its denotation is a function that takes a function A of type et and returns a function that takes a function B, also of type et, and returns a function that takes an entity x and returns 1 if and only if x satisfies both A and B.

$$\text{A} = \text{IS} = \lambda A_{et}.A$$
$$\iota = \lambda A_{et}.\begin{cases} a & A = (\lambda x_e.x = a) \\ \text{undefined} & \text{otherwise} \end{cases}$$
$$\text{WHO}_A = \lambda A_{et}.\lambda x_e.\iota(\lambda y.y = x \wedge A(x))$$
$$R_m = \lambda M_{(et)(et)}.\lambda A_{et}.\lambda x_e.M(A)(x) \wedge A(x)$$
$$\text{SOME} = \lambda A_{et}.\lambda B_{et}.\exists x.A(x) \wedge B(x)$$
$$\text{AND} = \lambda A_{et}.\lambda B_{et}.\lambda x_e.A(x) \wedge B(x)$$

Figure 1: Functions in the Lexicon

By marking words and syntactic constructions with lexical items, annotators indicate the underlying linguistic phenomena in the data. Furthermore, the formal foundation of this approach allows annotators to verify that the entailment relation (or lack thereof) that obtains between the textual forms of T and H also obtains between their respective semantic forms. This verification guarantees that the annotations are sufficient in the sense of providing enough information for recognizing the entailment relation based on the semantic abstraction. For example, consider the simple entailment *Dan sat and sang* $\Rightarrow$ *Dan sang* and assume annotations of *Dan* as a proper name, *sat* and *sang* as intransitive verbs and *and* as predicate conjunction. The formal model can be used to verify these annotations by constructing a proof as follows: for each model M:

$$\begin{aligned}
&[\![\textit{Dan } [\textit{sat } [\textit{and sang}]]\,]\!]^M && \\
&= ((\textsc{and}(\mathbf{sing}))(\mathbf{sit}))(\mathbf{dan}) && \text{analysis} \\
&= (((\lambda A_{et}.\lambda B_{et}.\lambda x_e.A(x) \wedge B(x))(\mathbf{sing}))(\mathbf{sit}))(\mathbf{dan}) && \text{def. of } \textsc{and} \\
&= \mathbf{sit}(\mathbf{dan}) \wedge \mathbf{sing}(\mathbf{dan}) && \text{func. app. to } \mathbf{sing}, \mathbf{sit} \text{ and } \mathbf{dan} \\
&\leq \mathbf{sing}(\mathbf{dan}) && \text{def. of } \wedge \\
&= [\![\textit{Dan sang}\,]\!]^M && \text{analysis}
\end{aligned}$$

3 Platform Architecture

The platform's architecture is based on a client-server model, as illustrated in Figure 2.

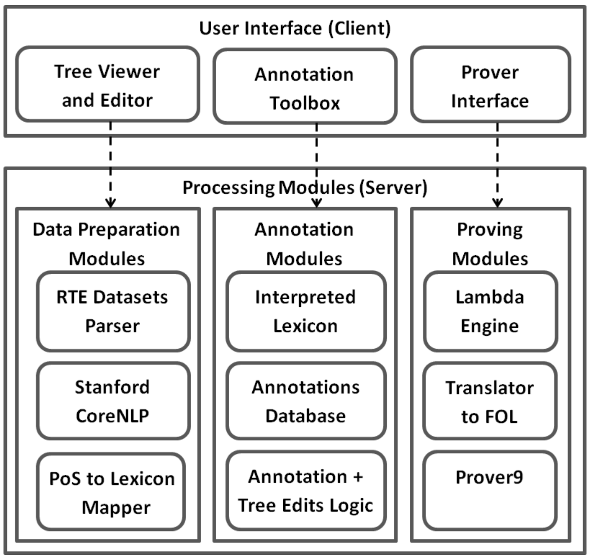

Figure 2: Platform Architecture

The user interface (UI) is implemented as a web-based client using Google Web Toolkit (Olson, 2007) and allows multiple annotators to access the RTE data, to annotate, and to substantiate their annotations. These operations are done by invoking corresponding remote procedure calls at the server side. Below we describe the system components as we go over the work-flow of annotating Example 1.

Data Preparation: We extract T-H pairs from the RTE datasets XML files and use the Stanford CoreNLP (Klein and Manning, 2003; Toutanova et al., 2003; de Marneffe et al., 2006) to parse each pair and to annotate it with part-of-speech tags.[4] Consequently, we apply a naive heuristic to map the PoS tags to the lexicon.[5] This process is called as part of the platform's installation and when annotators need to simplify the original RTE data in order to avoid syntactic/semantic phenomena that the semantic engine does not support. For example, the bare plural *short stories* is simplified to *some short stories* as otherwise the engine is unable to determine the quantification of this noun.

Annotation: The annotation is done by marking the tree-leaves with entries from the lexicon. For example, *short* is annotated as a restrictive modifier (*MR*) of the noun *stories*, and *contains* is annotated as a transitive verb (*V_2*). In addition, annotators manipulate the tree structure to fix parsing mistakes and to add leaves that mark semantic relations. For instance, a leaf that indicates the apposition between *the famous Bulgarian writer* and *Nikolai Haitov* is added and annotated as $\textsc{who}_A$. The server stores a list of all annotation actions. Figure 3 shows the tree-view, lexicon, prover and annotation history panels in the UI.

Proving: When annotating all leaves and manipulating the tree structures of T and H are done, the annotators use the prover interface to request a search for a proof that indicates that their annotations are substantiated. Firstly, the system uses lambda calculus reductions to create logical forms that represent the meanings of T and H in higher-order logic. At this stage, type errors may be reported due to erroneous parse-trees or annotations. In this case an annotator will fix the errors and re-run the proving step. Secondly, once all type errors are resolved, the higher-order representations are lowered to first order and Prover9 (McCune, 2010) is executed to search for a proof between the logical expressions of T and H.[6] The proofs are recorded in order to be included in the corpus release. Figure 4 shows the result of translating T and H to an input to Prover9.

4 Corpus Preparation

We have so far completed annotating 40 positive entailments based on data from RTE 1-4. The annotation is a work in progress, done by four Master students of Linguistics who are experts in the data and focus on entailments whose recognition relies on a mixture of appositive, restrictive or intersective modification. As we progress towards the compilation of a corpus of several hundred pairs, we extend the semantic model to support more inferences with less phenomena simplification.

[4] Version 1.3.4

[5] This heuristic is naive in the sense of not disambiguating verbs, adjectives and other types of terms according to their semantic features. It is meant to provide a starting point for the annotators to correct and fine-tune.

[6] Version 2009-11A

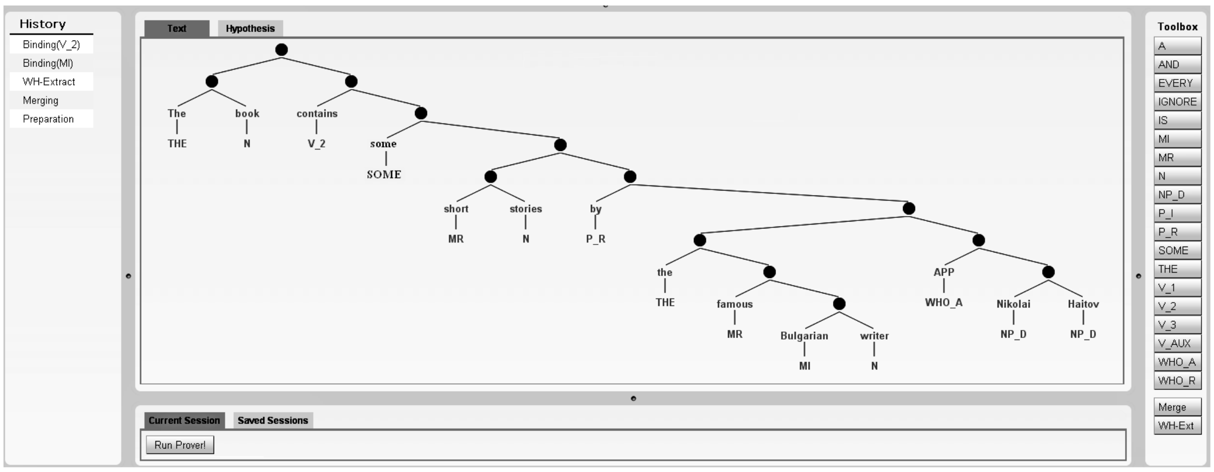

Figure 3: User Interface Panels: Annotation History, Tree-View, Prover Interface and Lexicon Toolbox

formulas(assumptions).
all x0 (((writer(x0) & bulgarian(x0)) & famous_writer_bulgarian(x0)) ↔ x0=c1).
all x0 (((stories(x0) & short_stories(x0)) & exists x1 (by_stories_short_stories(x1, x0) & (x1=c1 & x1=Nikolai_Haitov))) ↔ x0=c2). all x0 (book(x0) ↔ x0=c3).
contains(c2, c3).
end_of_list.

formulas(goals).
exists x0 (writer(x0) & x0=Nikolai_Haitov).
end_of_list.

Figure 4: Input for Theorem Prover

5 Conclusions

We introduced a new concept of an annotation platform which implements an *Annotating by Proving* approach. The platform is currently in use by annotators to indicate linguistic phenomena in entailment data and to provide logical proofs that substantiate their annotations. This method guarantees that the annotations constitute a complete description of the entailment relation and can serve as a gold-standard for entailment recognizers. The new corpus will be publicly available.

Acknowledgments

The work of Stavroula Alexandropoulou, Robert Grimm, Sophie Chesney, Pepijn Kokke, Benno Kruit, Kyriaki Neophytou, Antony Nguyen, Assaf Toledo and Yoad Winter was supported by a VICI grant number 277-80-002 by the Netherlands Organisation for Scientific Research (NWO).

References

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the IEEE / ACL 2006 Workshop on Spoken Language Technology*. The Stanford Natural Language Processing Group.

Danilo Giampiccolo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, and Elena Cabrio. 2008. The fourth pascal recognising textual entailment challenge. In *TAC 2008 Proceedings*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. ACL.

William McCune. 2010. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`.

Steven Douglas Olson. 2007. *Ajax on Java*. O'Reilly Media.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. ACL.