TruthTeller: Annotating Predicate Truth

Amnon Lotan
Department of Linguistics
Tel Aviv University
amnonlot@post.tau.ac.il

Asher Stern and **Ido Dagan**
Department of Computer Science
Bar Ilan University
astern7@gmail.com dagan@cs.biu.ac.il

Abstract

We propose a novel semantic annotation type of assigning truth values to predicate occurrences, and present TruthTeller, a standalone publicly-available tool that produces such annotations. TruthTeller integrates a range of semantic phenomena, such as negation, modality, presupposition, implicativity, and more, which were dealt only partly in previous works. Empirical evaluations against human annotations show satisfactory results and suggest the usefulness of this new type of tool for NLP.

1 Introduction

In a text, the action or relation denoted by every predicate can be seen as being either positively or negatively inferred from its sentence, or otherwise having an unknown truth status. Only in (3) below can we infer that Gal sold her shop, hence the positive *truth value* of the predicate *sell*, while according to (2) and (4) Gal did not sell it, hence the negative truth values, and in (1) we do not know if she sold it or not (the notations PT+, PT- and PT? denote truth states, defined in Subsection 2.3). Identifying these predicate truth values is an important subtask within many semantic processing scenarios, including various applications such as Question Answering (QA), Information Extraction (IE), paraphrasing and summarization. The following examples illustrate the phenomenon:

(1) *Gal made an attempt* $^{pt+}$ *to sell* $^{pt?}$ *her shop.*

(2) *Gal did* ***not*** *try* $^{pt-}$ *to sell* $^{pt-}$ *her shop* ***after*** *hearing* $^{pt+}$ *the offers.*

(3) ***Maybe*** *Gal wasn't smart* $^{pt?}$ *to sell* $^{pt+}$ *her shop.*

(4) *Gal wasn't smart* $^{pt-}$ ***enough*** *to sell* $^{pt-}$ *the shop* ***that*** *she had bought* $^{pt+}$.

Previous works addressed specific aspects of the truth detection problem: Nairn et al. (2006), and later MacCartney & Manning (2007; 2009), were the first to build paraphrasing and inference systems that combine negation (see *try* in (2)), modality (*smart* in (3)) and "natural logic", a recursive truth value calculus (*sell* in (1-3)); recently, Mausam et al. (2012) built an open IE system that identifies granulated variants of modality and conditions on predicates (*smart* in (3)); and Kiparsky & Kiparsky (1970) and Karttunen (1971; 2012) laid the ground work for factive and implicative entailment calculus (*sell* in (1-4)), as well as many generic constructions of *presupposition* (*hearing* in (2) is presupposed because it heads an adverbial clause and *bought* in (4) heads a finite relative clause), which, to our knowledge, have not yet been implemented computationally. Notice in the examples that presuppositions persist under negation, in questions and if-clauses, while entailments do not. In addition, there is a growing research line of negation and modality detection. See, for example, Morante & Daelemans (2012).

We present TruthTeller[1], a novel algorithm and system that identifies the truth value of each predicate in a given sentence. It annotates nodes in the text's dependency parse-tree via a combination of pattern-based annotation rules and a recursive algorithm based on natural logic. In the course of computing truth value, it also computes the implicativity/factivity signature of predicates, and their negation and modality to a basic degree, both of which are made available in the system output. It addresses and combines the aforementioned phenomena (see Section 2), many of which weren't dealt in previous systems.

TruthTeller is an open source and publicly available annotation tool, offers a relatively simple algebra for truth value computation, and is accompanied by a publicly available lexicon of over 1,700 implicative and factive predicates. Also, we provide an intuitive GUI for viewing and modifying the algorithm's annotation rules.

2 Annotation Types and Algorithm

This section summarizes the annotation algorithm (a detailed report is available with the system release). We perform the annotations over dependency parse trees, generated according to the Stanford Dependencies standard (de Marneffe and Manning, 2008). For all verbs, nouns and adjectives in a sentence's parse tree, we produce the following 4 annotation types, given in the order they are calculated, as described in the following subsections:

1. Predicate Implication Signature (*sig*) - describes the pattern by which the predicate entails or presupposes its complements, e.g., the verb *refuse* entails the negative of its complements: *Ed refused to pay* entails that *Ed didn't pay.*

2. Negation and Uncertainty (NU) - indicates whether the predicate is modified by an uncertainty modifier like *might, probably,* etc., or whether it's negated by *no, never* etc.

3. Clause-Truth (CT) - indicates whether the entire clause headed by the predicate is entailed by the complete sentence

4. Predicate Truth (PT) - indicates whether the predicate itself is entailed by the sentence, as defined below

[1]http://cs.biu.ac.il/~nlp/downloads/TruthTeller

Before presenting the detailed definitions and descriptions below, we give a high-level description of TruthTeller's algorithm, where each step relies on the results of its predecessor: a) every predicate in the parse tree is annotated with a predicate implication signature, identified by lexicon lookup; b) NU annotations are added, according to the presence of uncertainty modifiers (*maybe, might, etc.*) and negation modifies (*not, never, etc.*); c) predicates in certain presupposition constructions (e.g., adverbial clauses, WH arguments) are annotated with positive CT values; d) the parse tree is depth-first scanned, in order to compute both CT and PT annotations by the recursive effects of factives and implicatives; e) in conjunction with the previous step, relative clause constructions are identified and annotated with CT and PT.

Except for steps a) and d), all of the procedure is implemented as an ordered sequence of *annotation rule* applications. An annotation rule is a dependency parse tree template, possibly including variables, which assigns certain annotations to any parse tree node that matches against it. Step a) is implemented with signature lexicon lookups, and step d) is an algorithm implemented in code.

To illustrate this entire process, Figure 1 presents the annotation process of a simple sentence, step by step, resulting in TruthTeller's complete output, fully specified below. Most other examples in this paper show only partial annotations for brevity.

2.1 Predicate Implication Signature

Our system marks the signature of each predicate, as defined in Table 1. There, each signature has a left sign and a right sign. The *left* sign determines the *clause truth value* of the predicate's complements, when the predicate is in *positive* contexts (e.g., not negated), while the *right* sign applies in *negative* contexts (clause

#	Sig	Positive context example	Negative context example
1	+/-	Ed *managed* to escape ⇒ Ed escaped	Ed didn't *manage* to escape ⇒ Ed didn't escape
2	+/?	Ed was *forced* to sell ⇒ Ed sold	Ed wasn't *forced* to sell ⇒ *no entailments*
3	?/-	Ed was *allowed* to go ⇒ *no entailments*	Ed wasn't *allowed* to go ⇒ Ed didn't go
4	-/+	Ed *forgot* to pay ⇒ Ed didn't pay	Ed didn't *forget* to pay ⇒ Ed paid
5	-/?	Ed *refused* to fight ⇒ Ed didn't fight	Ed didn't *refuse* to fight ⇒ *no entailments*
6	?/+	Ed *hesitated* to ask ⇒ *no entailments*	Ed didn't *hesitate* to ask ⇒ Ed asked
7	+/+	Ed was *glad* to come ⇒ Ed came	Ed wasn't *glad* to come ⇒ Ed came
8	-/-	Ed *pretended* to pay ⇒ Ed didn't pay	Ed didn't *pretend* to pay ⇒ Ed didn't pay
9	?/?	Ed *wanted* to fly ⇒ *no entailments*	Ed didn't *want* to fly ⇒ *no entailments*

Table 1: Implication signatures, based on MacCartney & Manning (2009) and Karttunen (2012). The first six signatures are named implicatives, and the last three factive, counter factive and regular, respectively.

a) Annotate signatures via lexicons lookup

$$\textit{Gal wasn't allowed}^{?/-} \textit{ to come}^{?/?}$$

b) Annotate NU

$$\textit{Gal wasn't allowed}^{?/-,nu-} \textit{ to come}^{?/?,nu+}$$

c) Annotate CT to presupposition constructions

$$\textit{Gal wasn't allowed}^{?/-,nu-,ct+} \textit{ to come}^{?/?,nu+,ct+}$$

d) Recursive CT and PT annotation

$$\textit{Gal wasn't allowed}^{?/-,nu-,ct+,pt-} \textit{ to come}^{?/?,nu+,ct-,pt-}$$

e) Annotate CT and PT of relative clauses
(has no effect on this example)

$$\textit{Gal wasn't allowed}^{?/-,nu-,ct+,pt-} \textit{ to come}^{?/?,nu+,ct-,pt-}$$

Figure 1: An illustration of the annotation process

truth is defined in Subsection 2.3). See examples for both context types in the table. Each sign can be either + (positive), - (negative) or ? (unknown). The unknown sign signifies that the predicate does not entail its complements in any way.

Signatures are identified via lookup, using two lexicons, one for single-word predicates and the other for *verb+noun* phrasal verbs, e.g., *take the time to X*. Our single-word lexicon is similar to those used in (Nairn et al., 2006) and (Bar-Haim et al., 2007), but is far greater, holding over 1,700 entries, while each of the previous two has, to the best of our knowledge, less than 300 entries. It was built semi automatically, out of a kernel of 320 manually inspected predicates, which was then expanded with WordNet synonyms (Fellbaum, 1998). The second lexicon is the implicative phrasal verb lexicon of Karttunen (2012), adapted into our framework. The +/? implicative serves as the default signature for all unlisted predicates.

Signature is also sensitive to the type of the complement. Consider:

(6) *Ed forgot*$^{-/+}$ *to call*$^{pt-}$ *Joe*

(7) *Ed forgot*$^{+/+}$ *that he called*$^{pt+}$ *Joe*

Therefore, signatures are specified separately for finite and non finite complements of each predicate.

After the initial signature lookup, two annotation rules correct the signatures of +/+ factives modified by *enough* and *too*, into +/- and -/+, correspondingly, see Kiparsky & Kiparsky (1970). Compare:

(8) *Ed was mad*$^{+/+}$ *to go* ⇒ *Ed went*

(9) *Ed was* ***too*** *mad*$^{-/+}$ *to go* ⇒ *Ed didn't go*

In addition, we observed, like Karttunen (2012), that most verbs that have passive voice and the *into* preposition become +/? implicatives, e.g.,

(10) *Workers* ***were pushed / maddened / managed***$^{+/?}$ ***into*** *signing* ⇒ *They signed*

(11) *Workers* ***weren't pushed / maddened / managed***$^{+/?}$ ***into*** *signing* ⇒ *It is unknown whether they signed*

so we captured this construction in another rule.

2.2 Negation and Uncertainty (NU)

NU takes the values {NU+, NU-, NU?}, standing for non-negated certain actions, negated certain actions, and uncertain actions. The first NU rules match against a closed set of negation modifiers around the predicate, like *not, never, neither* etc. (see (2)), while later rules detect uncertainty modifiers, like *maybe, probably*, etc. Therefore, NU? takes precedence over NU-.

Many constructions of subject-negation, object-negation and "double negation" are accounted for in our rules, as in:

(12) ***Nobody*** *was seen*$^{nu-}$ *at the site*

(13) ***Almost nobody*** *was seen*$^{nu+}$ *at the site*

2.3 Clause Truth and Predicate Truth

Clause Truth (CT, denoted as *ct(p)*) corresponds to POLARITY of Nairn et al. (2006). It represents whether the clause headed by a predicate p is entailed by the sentence, contradicted or unknown, and thus takes three values {CT+, CT-, CT?}.

Predicate Truth (PT) (denoted as *pt(p)*) represents whether we can infer from the sentence that the action described by the predicate happened (or that its relation holds). It is defined as the binary product of NU and CT:

Definition 1. $PT = NU \cdot CT$

and takes analogous values: {PT+, PT-, PT?}. Intuitively, the product of two identical positive/negative values yields PT+, a positive and a negative yield PT-, and NU? or CT? always yield PT?. To illustrate these definitions, consider:

(14) *Meg* ***may*** *have slept*$^{ct+,pt?}$ ***after*** *eating*$^{ct+,pt+}$ *the meal Ed cooked*$^{ct+,pt+}$*,* ***while no*** *one was there*$^{ct+,pt-}$

After signatures and NU are annotated, CT and PT are calculated. At first, we apply a set of rules that annotate generic presupposition constructions with CT+. These include adverbial clauses opening with {*while, before, after, where, how come, because, since, owing to, though, despite, yet, therefore...*}, WH arguments (*who, which, whom, what*), and

$$ct(p) = \begin{cases} \text{CT+}: & p \text{ was already annotated by a presupposition rule} \\ ct(gov(p)): & p \text{ heads a relative clause} \\ compCT(p): & \text{otherwise, and } p \text{ is a complement} \\ \text{CT?}: & \text{otherwise (default)} \end{cases}$$

Figure 2: Formula of *ct(p)*, for any predicate p. $ct(gov(p))$ is the CT of p's governing predicate.

parataxis[2]. See for example the effects of *after* and *while* in (14).

Then, we apply the following recursive sequential procedure. The tree root always gets CT+ (see *slept* in (14)). The tree is then scanned downwards, predicate by predicate. At each one, we compute CT by the formula in Figure 2, as follows. First, we check if one of the aforementioned presupposition rules already matched the node. Second, if none matched, we apply to the node's entire subtree another set of rules that annotate each relative clause with the CT of its governing noun[3], $ct(gov(p))$ (see *failed* in (15)). Third, if no previous rule matched, and p is a complement of another predicate $gov(p)$, then *compCT(p)* is calculated, by the following logic: when $pt(gov(p))$ is PT+ or PT-, the corresponding left or right sign of $sig(gov(p))$ is copied. Otherwise, if $pt(gov(p)) =$ PT?, CT? is returned, except when the signature of $gov(p)$ is +/+ (or -/-) factive, which always yields CT+ (or CT-).

Third, if nothing applied to p, CT? is returned by default. Finally, PT is set, according to Definition 1.

To illustrate, consider these annotations:

(15) *Gal managed*$^{+/-,ct+,pt+}$ *a building*$^{+/?,ct+,pt+}$*, which Ginger failed*$^{-/+,ct+,pt+}$ *to sell*$^{+/?,ct-,pt-}$

First, *managed* gets CT+ as the tree root. Then, we get $compCT(building) =$ CT+, as the complement of *managed*$^{+/-,pt+}$. Next, a relative clause rule copies CT+ from *building* to *failed.*

[2]The placing of clauses or phrases one after another, without words to indicate coordination, as in "veni, vidi, vici" in contrast to "veni, vidi *and* vici".

[3]We also annotate nouns and adjectives as predicates in copular constructions, and in instances where nouns have complements.

Finally, $compCT(sell)$ = CT- is calculated, as the complement of $failed^{-/+,pt+}$.

3 Evaluation

To evaluate TruthTeller's accuracy, we sampled 25 sentences from each of the RTE5 and RTE6 Test datasets (Bentivogli et al., 2009; Bentivogli et al., 2010), widely used for textual inference benchmarks. In these 50 sentences, we manually annotated each predicate, 153 in total, forming a gold standard. As baseline, we report the most frequent value for each annotation. The results, in Table 2, show high accuracy for all types, reducing the baseline CT and PT errors by half. Furthermore, most of the remaining errors were due to parser errors, according to a manual error analysis we conducted.

The baseline for NU annotations shows that negations are scarce in these RTE datasets, which was also the case for CT- and PT- annotations. Thus, Table 2 mostly indicates TruthTeller's performance in distinguishing positive CT and PT annotations from unknown ones, the latter constituting ∽20% of the gold standard. To further assess CT- and PT- annotations we performed two targeted measurements. Precision for CT- and PT- was measured by manually judging the correctness of such annotations by TruthTeller, on a sample from RTE6 Test including 50 CT- and 124 PT- annotations. This test yielded 78% and 83% precision, respectively. PT- is more frequent as it is typically triggered by CT-, as well as by other constructions involving negation. Recall was estimated by employing a human annotator to go through the dataset and look for CT- and PT- gold standard annotations. The annotator identified 40 "CT-"s and 50 "PT-"s, out of which TruthTeller found 47.5% of the "CT-"s and 74% of the "PT-"s. In summary, TruthTeller's performance on our target PT annotations is quite satisfactory with 89% accuracy overall, having 83% precision and 74% recall estimates specifically for PT-.

Annotation	**TruthTeller**	**Baseline**
Signature	89.5%	81% (+/?)
NU	98%	97.3% (NU+)
CT	90.8%	78.4% (CT+)
PT	89%	77% (PT+)

Table 2: The accuracy measures for TruthTeller's 4 annotations. The right column gives the accuracy for the corresponding most-frequent baseline: {+/?, NU+, CT+, PT+}.

4 Conclusions and Future Work

We have presented TruthTeller, a novel algorithm and system that identifies truth values of predicates, the first such system to a) address or combine a wide variety of relevant grammatical constructions; b) be an open source annotation tool; c) address the truth value annotation task as an independent tool, which makes it possible for client systems to use its output, while previous works only embedded annotations in their task-specific systems; and d) annotate *unknown* truth values extensively and explicitly.

TruthTeller may be used for several purposes, such as inferring parts of a sentence from the whole and improving textual entailment (and contradiction) detection. It includes a novel, large and accurate, lexicon of predicate implication signatures.

While in this paper we evaluated the correctness of TruthTeller as an individual component, in the future we propose integrating it in a state-of-the-art RTE system and report its impact. One challenge in this scenario is having other system components interact with TruthTeller's decisions, possibly masking its effects. In addition, we plan to incorporate monotonicity calculations in the annotation process, like in MacCartney and Manning (2009).

5 Acknowledgements

This work was partially supported by the Israel Science Foundation grant 1112/08 and the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 287923 (EXCITEMENT).

We thank Roni Katzir and Fred Landman for useful discussions.

References

Roy Bar-Haim, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*, pages 871–876.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *Preproceedings of the Text Analysis Conference (TAC)*.

Luisa Bentivogli, Peter Clark, Ido Dagan, Hoa T. Dang, and Danilo Giampiccolo. 2010. The sixth PASCAL recognizing textual entailment challenge. In *The Text Analysis Conference (TAC 2010)*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *COLING Workshop on Cross-framework and Cross-domain Parser Evaluation.*

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. MIT Press.

Lauri Karttunen. 1971. Implicative verbs. *Language*, 47:340–358.

Lauri Karttunen. 2012. Simple and phrasal implicatives. In **SEM 2012*, pages 124–131.

P. Kiparsky and C. Kiparsky. 1970. Fact. In *Progress in Linguistics*, pages 143–173. The Hague: Mouton de Gruyter.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of ACL workshop on textual entailment and paraphrasing.*

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eighth International Conference on Computational Semantics (IWCS-8).*

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534.

Roser Morante and Walter Daelemans. 2012. Annotating modality and negation for a machine reading evaluation. In *CLEF (Online Working Notes/Labs/Workshop).*

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *In Proceedings of ICoS-5 (Inference in Computational Semantics).*