Syntactic testsuites and Textual Entailment Recognition

Paul Bedaride[*], **Claire Gardent**[+]

∗ INRIA/LORIA and Université Henri Poincaré
+ CNRS/LORIA
paul.bedaride@loria.fr claire.gardent@loria.fr

Abstract

We focus on textual entailments mediated by syntax and propose a new methodology to evaluate textual entailment recognition systems on such data. The main idea is to generate a syntactically annotated corpus of pairs of (non-)entailments and to use error mining to identify the most likely sources of errors. To illustrate the approach, we apply this methodology to the Afazio RTE system and show how it permits identifying the most likely sources of errors made by this system on a testsuite of 10 000 (non) entailment pairs.

1. Introduction

The Recognising Textual Entailment (RTE) challenge evaluates the ability of NLP systems to detect whether one sentence implies (*textually entails*) another. However, because it is constructed semi-automatically from real text, the data used for this challenge displays the full "NLP complexity": semantic construction but also anaphora resolution, temporal and spatial reasoning and reasoning based on lexical and general world knowledge. This makes it difficult to assess how well RTE systems deal with each of these processes and consequently, how far NLP is from providing them with a principled treatment.

In this paper, we focus on syntax based entailments that is, entailments such as (1) where entailment is mediated by syntax alone.

(1) a. John sends a book to Mary
→ John sends a book

b. John sends a book to Mary
→ A book is sent to Mary by John

c. John quickly sent a book to Mary
→ John's sending of a book to Mary was fast

To evaluate the capacity of RTE systems to deal with such cases, we propose a methodology that combines the automated construction of graduated, annotated, testsuites with a fine grained statistical data analysis of the system results that relies on testsuite annotations that are produced automatically.

This methodology differs from the traditional, corpus based analysis used in the RTE challenge in two main ways. First, it permits evaluating RTE systems on a specific entailment type, in this case, syntax based entailment. Second, the testsuite annotations and the statistical data analysis it supports, permit a precise identification of the most probable sources of errors.

We first sketch the method used for creating graduated testsuites for syntax-based entailment (Section 2.). We then evaluate an RTE system on this suite (Section 3.) . Finally, we show how error mining techniques which were originally developped for identifying errors in deeo computational grammars can be used to detect the most likely sources of RTE errors (Section 4.).

2. Creating syntactic testsuites

An RTE testsuite consists of pairs of sentences annotated with either TRUE or FALSE depending on whether the first sentence textually entails the other or not.

To build an RTE testsuite which focuses on entailments mediated by syntax alone, we use a technique inspired from template based surface realisation (Johnson et al., 2004). We start by manually specifying a set of syntactic patterns together with a lexicon linking patterns to strings. We then use this knowledge to build sentences and associate them with a meaning representation.

It is worth stressing that our aim is here to illustrate the usefulness of the proposed evaluation methodology not to provide an extensive coverage of the possible syntactic variations on a given content.

2.1. Syntactic patterns

We provide an XML specification of the possible syntactic patterns acceptable for various verb types (e.g., transitive, intransitive) of English. A syntactic pattern is a sequence of words and syntactic categories. It is furthermore annotated with a subcategorisation type and with one or more syntactic tags describing the specific syntactic constructs involved in this pattern such as active (A) or passive (P) verb form, relativised argument (R) and verbal (V) *vs.* nominal (N) predicate.

For instance, the following patterns describe a string of the form NP-V-Prep-NP where the verb has subcategorisation type nVnPn i.e., ditransitive. The first pattern describes a sentence where the verb is in the active form (V-A e.g., "John sent a book to Mary"); the second pattern describes a sentence where the verb is in the passive form (V-P-PP0 e.g., "A book is sent to Mary by John"); and the third pattern describes an active voice sentence with a relativised subject (e.g., "The man who sent a book to Mary ...").

```
<Const id="nVnPn" type="V-A">
  <NP id="0"/> <Verb /> <NP id="1"/>
  <PP id="0"/> <NP id="2"/>
</Const>

<Const id="nVnPn" type="V-P-PP0">
  <NP id="1"/> is <PPVerb />
  <PP id="0"/> <NP id="2"/>
  by <NP id="0"/>
</Const>

<Const id="nVnPn" type="V-R0-A-PP0"
       relPro="0">
  <NP id="0"/> who <Verb /> <NP id="1"/>
  <PP id="0"/> <NP id="2"/>,
</Const>
```

2.2. A Base Lexicon for the Syntax/Semantics interface

To build sentences out of the syntactic patterns, we specify a base lexicon which lists for each predicate, the corresponding verb and noun forms, the verb subcategorisation type, the semantic type of its arguments and the mapping between syntactic and semantic arguments.

For instance, the entry for the ”send” predicate is as follows.

```
<Set pred="send">
  <Parts>
    <Verb>sends</Verb>
    <PPVerb>sent</PPVerb>
    <Noun>sending</Noun>
    <Arg n="0" id="Person"/>
    <Arg n="1" id="Object"/>
    <Arg n="2" id="Person"/>
  </Parts>
  <Dists>
    <Dist constfam="possNpn"
          dist="01" fill="of" />
    <Dist constfam="nVn"
          dist="01" />
    <Dist constfam="nVnPn"
          dist="012" fill="to" />
  </Dists>
</Set>
```

This says that the predicate “send” can be realised by the verb forms *sent* or *sends* or by the noun form *sending*; that a sending event involves three participants named 0, 1 and 2 whose ontological types are Person, Object and Person respectively; and that three possible syntactic patterns and syntax-to-semantic mappings are possible namely, possNpn (*John's sending of a book*), nVn with distribution 01 (e.g., *John sent a book*) and nVnpn with preposition filler “to” and distribution 012 (e.g., *John sent a book to Mary*). The distribution shows the linear order of the semantic arguments realisation in the string.

Note that because the lexicon refers to syntactic families, each `Dist` specification may in fact license several syntactic patterns namely, all the syntactic patterns associated with the family referred to by the `Dist` element. For instance, given the three syntactic patterns listed in section 2.1. for the `nVnpn` verb type, the `send` predicate will licence the production of an active voice variant, a passive voice one and a relativised subject one.

2.3. Linking strings to semantic types

The information described so far refers to the syntax and semantics of predicative structures. To produce sentences out of these structures, information is also needed about the strings realising the predicates and their arguments. To this end, we specify for each ontological type referred to in the predicate specification, one or more example strings that are typical representatives of that type. For instance, the “Person” and “Object” type will be associated with example strings as follows.

```
<SemArgs>
   <SemArg id="Person">
     <Arg>John</Arg>
     <Arg>Mary</Arg>
     <Arg>Mark</Arg>
     <Arg>Jane</Arg>
     <Arg>Kevin</Arg>
     <Arg>The man</Arg>
   </SemArg>
   <SemArg id="Object">
     <Arg>a book</Arg>
     <Arg>a fork</Arg>
     <Arg>a spoon</Arg>
   </SemArg>
</SemArgs>
```

2.4. Generating a testsuite of textual entailments mediated by syntax

The testsuite is generated in two steps. First, sentences are generated. Second, sentences are associated with a semantic representation and the entailment value between the two sentences is determined by comparing their associated semantics.

2.4.1. Generating sentences

For a given predicate, we generate the set of strings that can be associated with this predicate given the constraints stated in the base lexicon and the available syntactic patterns. For instance, given the `send` predicate and the above lexical and syntactic specifications, the following sentences/NPs will be produced:

(2) a. John sends a book

b. John sends a book to Mary

c. The man who sends a book to Mary

d. John's sending of a book

Complex sentences involving more than one clauses are produced by dedicated procedures dealing with e.g., relative clauses.

2.4.2. Generating testsuite items

A testsuite item consists of two sentences (generated as described above), a truth value (false or true) indicating whether the entailment holds and a set of syntactic tags associated with each of the two sentences e.g.,

T: John's sending of a book to Mary was fast
`send-{(N,-,Poss), (N,-,PP0)}`
H: John sent a book
`send-{(V,A,-)}`
E: True

To construct theses testsuite items, we start by associating each sentence with a semantic representation as follows. Each argument is associated with a triplet of the form $(predicate, role, arg)$ where $predicate$ is the value of the `pred` attribute, $role$ is the `n` id of the argument `Arg` and arg is the `Arg` value (picked among the corresponding `SemArgs`). For instance, the semantics of ”John sends a mail to Mary” and of ”John who send a mail to Mary” is

$$(send, 0, John), (send, 1, mail), (send, 2, Mary)$$

In other words, the meaning representation obtained captures basic predicate/argument relationship whereby other semantic phenomena such as e.g., quantification are not dealt with. Clearly such phenomena have an impact on entailment and should be catered for at some point. However since we are mostly concerned with setting up a methodology which supports the evaluation of syntax based entailment, we leave their handling for further research.

A testsuite item $\langle S_1, S_2\rangle$ with semantics Sem_1 and Sem_2 will be annotated with TRUE if $Sem_1 \subset Sem_2$ and with FALSE otherwise.

The syntactic annotations labelling each sentence are collected from the syntactic patterns used to construct the sentence.

3. Evaluating an RTE system

Although the testsuite building tool described in the previous section is far from covering all possible syntactic variations of a given sentence, it provides a good starting point for evaluating the ability of RTE systems to handle such variation. To illustrate this, we evaluate the Afazio RTE system on a testsuite created using that tool.

3.1. The Afazio RTE system

Similarly to the Nutcracker system (Curran et al., 2007), the Afazio RTE system combines a statistical parser (the Stanford parser (Klein and Manning, 2003)) with a symbolic semantic component. First, a system of cascaded rewrite modules is used to rewrite the output of the parser into a “normalised” semantic representation intended to abstract away from surface differences and assign paraphrases the same representation (Bedaride and Gardent, 2009a; Bedaride and Gardent, 2009b). Special emphasis is placed on capturing syntax based equivalences such as syntactic (e.g., active/passive) variations, redistributions and noun/verb variants. Next, automated reasoning is used to check entailment.

3.2. The evaluation testsuite

Using the methodology described in section 2.4., we built an entailment testsuite where entailment is mediated purely by syntax. The resulting testsuite is graduated in that it contains cases of varying syntactic and semantic type. More specifically, its composition can be summarised as follows:

- It consists of 10 testsuites of 1 000 test items each.
- Each testsuite contains an equal distribution of true and false entailments.
- Each test suite contains sentences made up of up to three clauses.
- The predicates (verbs and nouns) used by each test suite have one of 10 distinct subcategorisation type.
- Each test suite uses a different combination of subcategorisation types e.g., INTRANSITIVE-VERB, TRANSITIVE-VERB, DITRANSITIVE vs. INTRANSITIVE-VERB-WITH-PLURAL-SUBJECT, TRANSITIVE-VERB, DITRANSITIVE.

3.3. Results

The results obtained by the Afazio system on this testsuite are given in the following table:

	True	False	Total
Correct	2445	4872	7317
Incorrect	2482	33	2515
Errors	73	95	168
Total	5000	5000	10000

True and False are the gold values for entailment (True indicates a sentence pair such that the first sentence textually entails the other, False a sentence pair where no such entailment holds). Correct and Incorrect give the system’s behaviour with respect to the gold. In particular, Incorrect indicates mismatches between the system result and the gold annotation. Finally, errors are cases where automated reasoners fail to return an answer (since first order logic is only semi decidable, there is never a garantee that automated reasoners succeed in determining whether or not entailment holds).

Although the overall accuracy is reasonable (73.2 %), false negatives (that is, incorrect system answers on true entailment or equivalently, entailments that are labelled as non entailments by the system) are numerous. To get a clearer picture of which linguistic phenomena are ill handled we perform error mining on the afazio results as explained below.

4. Finding the source of errors

The annotations contained in the automatically constructed testsuite allow us to caracterise the most important sources of failures. To this end, we use error mining techniques that were developed by the parsing community to identify the most likely sources of errors in manually specified, deep grammars. In specific, we use (Sagot and de La Clergerie, 2006)’s suspicion rate to compute the probability that a given syntactic tag pair (cf. section 2.) is responsible for an RTE detection failure. The tags with highest suspicion rate indicate which syntactic phenomena often cooccurs with such failure.

For each testsuite item (T,H), we store all tag pairs (TAG_H, TAG_T) such that TAG_T and TAG_H are associated with the same predicate but TAG_T occurs in T and TAG_H in H.

Given a testset and a subset of incorrectly recognised entailments, the supicion rate of each tag pair is computed usin a fix point algorithm. The suspicion rate of a form f after n iteration is written S_f^n. (Sagot and de La Clergerie, 2006) propose differents rankings with several measure functions M_f. The simplest one is $M_f = S_f^n$ that get the most certains errors. Another one takes into account the number of pairs $|\mathcal{O}_f|$ where a form f occurs. Instead, we prefer to consider the number of errors pairs $|\mathcal{O}_f^{err}|$. The reason for this is that sometimes a form has a low suspicion rate, occurs in a couple of error pairs but in a lot of pairs, and the score will be high even if solving the form's problem will not really improve the system. The first corrected measure function is $M_f = S_f * |\mathcal{O}_f|$ that calculate the expected value of the gain (in term of error pairs that become ok) if we correct problems due to form f. The last one, which we used to produce our results, is a balance between the two others functions $M_f = S_f * log(|\mathcal{O}_f|)$.

4.1. How to mine for errors in RTE results

There are several ways to mine for errors given the method just sketched and the data we have available. A first option that comes to mind is to take the whole corpus and to consider as a failure all mismatches between system and gold answers.

A drawback of this method is that it fails to differentiate beween false positives and false negatives. In RTE, false positives are cases where the system labels a true entailment as false. Conversely, false negatives occur when the systems labels a non entailment as true. Intuitively, these are very different types of errors. Moreover, given a certain type of RTE system and a certain type of data, false positives might be more frequent than false negative or vice versa. For instance, given an RTE system based on word overlap and testsuites such as ours, which (because the lexicon is restricted) exhibit a higher than average overlap between the two sentences, false positives will be higher than false negatives. That is, an RTE system might be biased in such a way that it produces more false positives than false negatives or vice versa.

We therefore chose to perform error mining separately for false positives and for false negatives. Given this, to compute the suspicion rate of each pairs of syntactic tags, we compare, for each such pair, its number of occurrences in false positives (false negatives) cases with its total number of occurrences in non entailment (entailment) pairs.

4.2. Results analysis

Table (1) shows the results of error mining for false negatives. The first line indicates that sentence pairs where a given predicate is realised as a verb (V) in the passive (P) voice with one relativised (R) argument in the hypothesis (H) but as an active (A) voice verb (V) involved in a coordination (C) in the text (T), has 61,02% of chance of yielding a false negative. Further, there are 65 such cases in the testsuite. Hence if the system can be corrected for this type of errors the net gain will be of 37.22 corrected pairs.

More generally, looking at the top rows, we observe that N/V alternations occur frequently (line 2, 3, 5 and 6) and that in V/V configurations, C (i.e., coordination) recurs (lines 1 and 4). Improving the system handling of noun/verb alternations and of coordinations is therefore likely to improve overall performance.

Looking now at the figures given by mining for errors on false positives (Table 2), we observe a really low error rate compared to the results for false negatives. This is due of course, to the much smaller amount of relevant data (33/4906) to be worked with. Nonetheless, the figure clearly show that false positives frequently occurs whenever a relativised (R) or prepositional (PP0) argument is used. This is likely related to incorrect PP attachment and relative pronoun resolution, two processes with a strong impact on the resulting Predicate/Argument representations.

Rank	Tag pair		S_f^{10}	$\frac{\#errors}{\#pairs}$	M_f^{err}
	H	T			
1	V P R	V A C	0.6102	61/65	2.547
2	V A R	N PP0	0.4038	349/539	2.5401
3	N PP0	V A R	0.3828	291/378	2.2717
4	V A R	V A C	0.5397	51/58	2.1913
5	V A	N	0.7868	12/12	1.9552
6	V P	N	0.7848	11/11	1.8818
7	N PP0	N PP0	0.3207	245/486	1.9837
8	V A R	N	0.6697	13/14	1.7675
9	N PP0	V A	0.4093	63/102	1.8929
10	N PP0	V P R	0.3295	150/214	1.768

Table 1: Error minning results on false negatives

Rank	Tag pair		S_f^{10}	$\frac{\#errors}{\#pairs}$	M_f^{err}
	H	T			
1	V P R	V P	0.0203	5/246	0.1119
2	N PP0	N PP0	0.0067	9/456	0.0407
3	V A	V P	0.0202	2/66	0.0846
4	N PP0	N S	0.0058	3/150	0.029
5	V A PP0	V A R	0.0039	4/233	0.0214
6	V P R	V P R	0.0025	9/1213	0.0176
7	V A	V A R	0.0077	2/174	0.0395
8	V A PP0	V P PP0	0.0048	3/202	0.0257
9	V A R	V A R	0.0024	8/761	0.016
10	V P R	N S	0.0069	2/69	0.0292

Table 2: Error minning results on false positives

5. Conclusion

The development of a linguistically principled treatment of the RTE task requires a clear understanding of the strength and weaknesses of RTE systems with respect to the various types of reasoning involved. We presented an evaluation framework which focuses on the simplest of these reasoning namely syntax based reasoning. As the results show, there is room for improvment even on that most basic level. The main contribution of this paper is the specification of an evaluation methodology which permits a focused evaluation of RTE systems. Future work will concentrate on the following points.

First, we plan to use the existing framework to compare the Afazio RTE system with other RTE systems and in particular, with a system integrating an off the shelf Semantic Role Labeller (SRL). Indeed SRLs aim to capture syntax based equivalences. We intend to compare the Afazio RTE systems which integrates a symbolic SRL module with an off the shelf SRL system extended with a semantic reasoning module similar to that used in the Afazio RTE system.

Second, we will investigate in how far surface realisation systems (systems that generate from a conceptual representation one or more string(s) verbalising that representation) can be put to work to automatically create graduated syntactic testsuites. We plan in particular, to use the GenI surface realiser (Gardent and Kow, 2005) to generate such suites. Because GenI is based on a large scale grammar which describes both syntax and semantics, using it to create RTE testsuites should help to overcome the semantic and syntactic limitations of the testsuite creation tool sketched in this abstract.

6. References

P. Bedaride and C. Gardent. 2009a. Normalising semantics : a framework and an experiment. In *IWCS 2009 (International Conference on Computational Semantics)*, Tilburg, The Netherlands.

P. Bedaride and C. Gardent. 2009b. Noun/verb entailment. In *4th Language and Technology Conference*, Poznan, Poland.

J.R. Curran, S. Clark, and J. Bos. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic.

C. Gardent and E. Kow. 2005. Generating and selecting grammatical paraphrases. *ENLG*, Aug.

W. Lewis Johnson, P. Rizzo, W. Bosma, S. Kole, and M. Ghijsen. 2004. Generating socially appropriate tutorial dialog. In *Workshop on Affective dialog systems*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, Morristown, NJ, USA. Association for Computational Linguistics.

B. Sagot and E. de La Clergerie. 2006. Error mining in parsing results. In *Proceedings of ACL-CoLing 06*, pages 329–336, Sydney, Australie.