Identifying Speculation from Syntax

Abstract

The RelEx semantic relation extractor is a dependency parser that generates feature markup identifying hypothetical and speculative statements, questions and clauses. These are identified by means of fixed, linguist-generated rules applied to syntactically parsed sentences. By using purely syntactic information, obtained from single sentences, this approach provides a baseline for extracting the semantic content of speculative sentences. This short note reviews the operation of the RelEx system, focusing on how it identifies of hypothetical and speculative statements.

1 Overview

The RelEx semantic relation extractor (Ross et al., 2005) is an open-source dependency parser built on top of the CMU Link Grammar Parser (Sleator and Temperley., 1991; Sleator and Temperley, 1993). Starting with a parse generated by Link Grammar, it applies a set of rules or graph transformations to generate commonly recognized dependencies (such as subject, object, adjective and noun modifiers, *etc.*) and feature tags (such as noun number, verb tense, *etc.*). At this level, it is roughly comparable to other dependency parsers, such as the Stanford parser (Marie-Catherine de Marneffe and Manning, 2006; de Marneffe and Manning, 2008) or Dekang Lin's MiniPar (Lin, 1998); indeed, it has an explicit Stanford-parser compatibility mode.

Aside from these basic functions, it also generates somewhat more semantically oriented feature and relation markup, which is not found in other parsers. This includes basic entity extraction; identification of times, dates, distances and units; identification of the query type, and the queried variable in a question; of the comparison variable in comparative statements and questions; and the identification of speculative and hypothetical statements and questions. All of this is done purely by means of a set of rules applied to a graph representation of the parsed sentence. Insofar as this information is obtained by purely syntactic methods, scoped to single, grammatically well-structured sentences, it should provide a baseline for what is possible, or could be expected from more sophisticated, semantic approaches.

Hypothetical statements occur in several forms: in statements about the future, in assertive questions, and as many dependent clauses. So, for example, predictions about the future are always hypothetical. In the sentence "*The dog will eat it*", the verb "*eat*" gets marked as being hypothetical – the dog hasn't eaten it yet, and there are no assurances that it will happen – only that it could happen. In this case, an imperative or future tense verb is sufficient to indicate the speculative nature of the statement. RelEx designates hypothetical or speculative statements by means of a feature tag – a single bit of markup, HYP, that is used to tag the verb making the hypothetical assertion. The generated output is thus:

```
_subj(eat, dog)
_obj(eat, it)
tense(eat, imperative)
HYP(eat, T)
```

Here, `_subj` and `_obj` are standard subject and object dependencies; as usual, the head word

is always listed first. The `tense` and `HYP` are feature tags: they identify a property, naming the word as the first argument, and the property value as second. Here, `T`, for 'true', simply means that the property is present. This listing, and the ones to follow, does not show all of the RelEx output; feature markup for part of speech, noun number, gender, tense, and other tags will be omitted except as relevant to the examples.

Speculation may be explicit or implicit: so "*It might rain*" is explicitly speculative, while "*It will rain*" seems to be very certain of itself, but is speculation none-the-less. All assertions about the future are inherently hypothetical until the future has become the past. Thus, "*It might rain*" generates:

```
_subj(rain, it)
_to-do(might, rain)
tense(rain, infinitive)
HYP(rain, T)
```

while "*It will rain*" generates:

```
_subj(rain, it)
tense(rain, future)
HYP(rain, T)
```

In both cases, the relation `_subj(rain, it)` is used for the copula; the relation `_to-do(might, rain)` notes the explicitly speculative nature of the copula.

Assertive questions are always hypothetical. For example, the question "*Fred is dead?*" might validly get an affirmative or a negative reply; Fred's being dead is hypothetical. Thus:

```
_predadj(Fred, dead)
TRUTH-QUERY(dead, T)
HYP(dead, T)
```

indicates that "deadness" is uncertain – if it was certain, there'd be no point in asking a question. The tag `TRUTH-QUERY` indicates not only that the sentence was a question, but that the truth of the assertion is being queried about. (The relation `_predadj`, the predicative adjective, is used here instead of `_subj(dead, Fred)`, to distinguish copular statements such as "*Mr. Smith is late*" from "*The late Mr. Smith*", the later generating `_adj(Smith, late)`).

Seemingly factual statements can be hypothetical as well. For example, in the sentence "*John says you like swimming*", the words "*like*" and "*swimming*" both get marked as hypothetical. "*Like*" is hypothetical, because John may be lying, or John may be in error about your likes and dislikes. "*Swimming*" is hypothetical, because you may indeed like something, just not swimming. Therefore, the claim that "*you like swimming*", appearing as a clause, is speculative.

```
_subj(swim, you)
_to-do(like, swim)
tense(swim, progressive)
HYP(swim, T)
HYP(like, T)
```

This example demonstrates the relation `_to-do`, which indicates a clausal complement. In general, both clausal complements, and adjectival complements (`_to-be`) indicate hypothetical statements; this will be seen explicitly in the rules.

The next section reviews the the operation of the RelEx rule engine, together with a discussion of some of the rules used to identify hypotheticals.

2 The Rule Engine

RelEx is a rule engine that extracts semantic content from syntactic constructions. Specifically, it represents syntactic relations as a graph, and then applies a sequence of rules to enlarge, prune, or transform the graph. This section reviews this mechanism; the next examines the rules used to identify speculative/hypothetical statements.

RelEx generates feature and dependency grammar markup by applying a sequence of pattern-matching rules to a labeled directed graph. An example of labeled directed graphs can be found in the hierarchical file systems used by most operating systems. In a file system, a directory listing shows either files or directories, both having names. One may move to a sub-directory, again containing named files or named directories. The graph is *not* acyclic; most hierarchical file systems allow '*links*', which point to directories or files in other locations. Thus, although one can choose to think of the structure as a hierarchical tree with links connecting sub-trees every

which-way, the proper way of conceptualizing is to think of a general graph, with individual edges being labeled arrows.

In retrospect, it appears that a better design point would have been to use a hypergraph for the data structure, as many relations are more easily expressed, and operations more easily performed, on hypergraphs. The OpenCog project (Hart and Goertzel, 2008; Goertzel, 2009), for which RelEx provides the NLP front-end, uses, at its core, a full hypergraph representation for its data.

The graph is initially populated with Link Grammar parse data. Link Grammar parses result in non-directional, labeled links between pairs of words (which may or may not be immediate neighbors). For example:

```
+-Ds-+--Ss--+--I-+-Ox+
|    |       |    |   |
the dog.n will.v eat it
```

The two words '*dog will*' and the connector `Ss` are imported into RelEx as shown in figure 1. The strings *dog* and *will* appear in the "subdirectories" named `str`. The previous and next words appear in "subdirectories" labeled `PREV` and `NEXT`. Additional relations connecting these words to the Link Grammar connector `Ss` are shown.

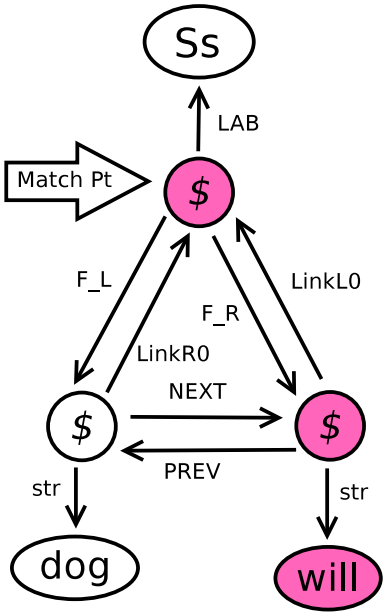
The graph is transformed by applying a sequence of rules, pattern-matching to an antecedent, and creating the consequent. That is, each rule has the form of '`if`*(antecedent)* `then` *consequent*', where the antecedent and consequent are both (sub-)graphs themselves. The rule engine searches out matching subgraphs, and creates the consequent when a match is found. An example of applying a rule consequent is shown in figure 2.

3 Identifying Hypotheticals

As of the current version (1.3.0), RelEx has twelve rules for identifying hypothetical or speculative verbs in clauses; these rules treat six classes of sentences. This section reviews these rules.

One of the rules for tagging assertive questions ("*Fred is dead?*") as hypothetical is the following:

Figure 1: Relation Graph

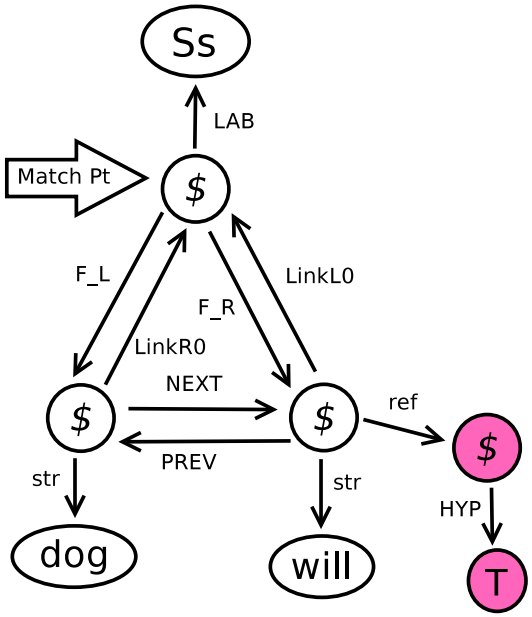


This figure illustrates a match for the rule antecedent `F_R str = will`. Starting at the indicated match point, there is an arrow labeled `F_R` pointing at a node with another arrow labeled `str` which points at a leaf node holding the string *'will'*.

```
if
  <LAB> = \Qd\.*
  <F_L str> = LEFT-WALL
then
  <F_R head-word ref TRUTH-QUERY> = T
  <F_R head-word ref HYP> = T
```

This rule looks for a Link Grammar `Qd` link attached to the `LEFT-WALL`. In the Link Grammar, the `LEFT-WALL` is the pseudo-word preceding the first word of the sentence; the `Qd` link is one of several commonly used when identifying questions of a certain form. If this pattern is found, then the head-word of the sentence is tagged with both the truth-query and hypothetical flags. The head-word was previously identified by an earlier sequence of rules; typically, it is the primary verb in the sentence. For the example "*Fred is dead?*", "*dead*" was previously identified as the head-word; the above rule merely confirms that the sentence was a declarative question, and then tags "*dead*" appropriately. A second rule, very similar to this, is used to tag a Link

Figure 2: Edited Graph



This figure illustrates the result of applying the rule consequent `F_R ref HYP = T` at the indicated match point. Two arrows, labeled `ref` and `HYP` have been created, the last arrow pointing at the leaf node `T`.

Grammar parse variant for copular declarative questions. It should be clear here that Link Grammar performed the "hard work" of determining that a `Qd` link is syntactically appropriate for the parse; this rule reveals the semantic content associated with this link. It works because the `Qd` link, when attached to the left wall, is always associated with truth query questions (which thus are hypothetical).

Questions that ask about quantity, or, more generally, inquire about a putative property of an object or situation are deemed hypothetical. The core semantic idea here is that if its putative, then its hypothetical. These are handled by three closely related rules. One is

```
if
  <LAB> = \Dm.*
  <F_L str> = how_much|how_many
then
  <F_L ref name> = _$qVar
  <F_L ref QUERY-TYPE> = how_much
  <F_R head-word ref HYP> = T
```

which is used to tag a question such as "*How much coffee did you drink?*". Here, the question is asking for a quantity, thus a query variable is identified, as well as the type of the query. The question is accusative, and thus implicitly speculative: you may not have drank any coffee at all. The resulting output is

```
_subj(drink, you)
_obj(drink, coffee)
_quantity(coffee, _$qVar)
QUERY-TYPE(_$qVar, how_much)
HYP(drink, T)
```

The above rule keyed off of a Link Grammar `Dm` link, used to identify determiners for uncountable (mass) nouns, together with previous rules that were used to identify the *'how_much*' polyword. Both the determiner, and one of the two polywords must be explicitly present in order for the head-word ('*drink*' in this example) to be labeled as hypothetical.

Not all questions in this class make use of a determiner; more generally, these employ adverbial constructions, such as "*How quickly did you read the book?*", "*How tall is that building?*", or adjectives playing an adverbial role: "*How safe is that vehicle?*". Here, "*quickly*" and "*tall*" are the adverbial modifiers; "*safe*" is an adjective acting as an adverbial modifiers; these are connected with Link Grammar `EE` or `EA` links. The questions are inherently speculative or putative: the vehicle may not be safe at all.

Note also that the above rule also determined a query variable, and the type of the query. A distinct rule determined that the query variable is for a quantity of coffee. In general, all questions in this class will not only have the head-verb tagged as hypothetical, but will also result in a query variable being identified. These are all presented at the output in the form of dependencies; these dependencies are not strictly syntactic in nature, because they introduce nodes that are not present as words in the input sentence. This should be contrasted to the output of purely syntactic parsers, such as MiniPar or the Stanford parser, which generate no such output. Roughly speaking, RelEx attempts to get at some of the deeper structures behind the surface, such as those discussed in dependency frameworks like Mel'čuk's Meaning-

Text Theory (Mel'cuk and Polguere, 1987; Steele, 1990).

Another class of speculative questions ask for a perpetrator: "*Who ate my lunch?*" As before, the accusative nature of the question renders it speculative: was it really eaten? The rule for identifying these is simple:

```
if
  <F_R ref QUERY-TYPE> = who
then
  <F_R head-word ref HYP> = T
```

the hard work having been the previous identification of the form of the question. The output for "*Who ate my lunch?*" is:

```
_obj(eat, lunch)
_poss(lunch, me)
_subj(eat, _$qVar)
QUERY-TYPE(_$qVar, who)
HYP(eat, T)
```

Another class of sentences are those with adjectival or clausal complements; these are tagged as hypothetical. For example: "*Mary says the rose smelled sweet*" results in

```
_subj(say, Mary)
_that(say, smell)
_subj(smell, rose)
_to-be(smell, sweet)
HYP(smell, T)
HYP(sweet, T)
```

Here, the `_to-be` relation identifies the copular construction of the clause. RelEx treats all such clausal constructions as inherently speculative: this seems semantically reasonable, as there is always a claimant who may not be accurately reporting the facts. The rule for this tagging is trivial:

```
if
  <_to-be> != %
then
  <_to-be HYP> = T
```

which searches for a subgraph where the labeled directed edge `_to-be` exists (is not empty); if this edge exists, then what it is pointing at is tagged as hypothetical. The heavy lifting was done by prior rules that identified the clause itself. Observe also that previous rules identified an implicit `_that` relation in the sentence: "*Mary says <that> ...*". In English, one may omit the word "*that*" in such constructions; however, it remains implicitly present as a dependency relation between the main sentence and the dependent clause. In a similar vein, a distinct rule marks the clause following an explicit subordinating "*that*" as hypothetical: *"I assumed that we would go*" results in the markup `HYP(go)`.

Semantic and syntactic dependencies do not always correspond; they may be inverted. A fourth rule is used to tag clauses subordinate to "*if*": "*I will do it if you say so*". The head word of the main clause is "*do*", it is tagged as hypothetical. The syntactically dependent clause is "*if you say so*". Unlike the previous examples, it is not the dependent clause that is hypothetical; but instead, the primary clause is. "*I will do it*" is a promise, but it depends on your saying so; the semantic dependency runs in the opposite direction from the grammatical dependency. The "*doing*" remains hypothetical precisely because it is a semantic dependent. But, as before, this inverted usage is readily visible in the syntactic structure of the sentence. The RelEx rule is nearly trivial:

```
if
  <if> != %
then
  <if HYP> = T
```

the semantically dependent "*I will do it*" having been identified as the target of "*if*" by previous rules.

That last class of expressions that are marked as hypothetical are negative assertions; for example: "*John wasn't there*" or "*that isn't the case*". The linguistic reasoning to support this is not well-grounded, but is based on the notion that such negative assertions are generated as replies to speculative questions, and thus are inherently argumentative, inheriting the speculative nature of the question. A proper study, based on corpus linguistics principles, might clarify whether such tagging is indeed semantically appropriate. More importantly, this case illustrates the limits of a purely syntactic approach to identification of speculative/hypothetical statements. The speculative nature of a negative assertion is no longer readily apparent from its grammatical construction; the context of the surrounding paragraphs or dialog provides a semantic backdrop that is necessary to prop-

erly interpret the remark. Syntactic parsing works precisely because the most frequent, "common-sense" meaning is encoded in a fixed way in sentence structure. Syntactic parsing breaks down whenever the speaker is "playing games" with the words, encoding a meaning that runs counter to the usual "common-sense" structure of a sentence. In the vast majority of speech and writing, the direct, common-sense interpretation of sentence structure is used by the speaker, and is assumed by the listeners. For negative assertions, the syntax no longer suggests a relatively unambiguous meaning; using simple rules operating on single sentences is no longer a productive way of extracting the semantic content.

4 Conclusions

A fair amount of speculative content can be identified purely from the syntactic structure of isolated English sentences, without reference to surrounding context. Such extraction is straightforward when one has access to a dependency parse of a sentence, and a rule engine that can identify the appropriate patterns. As always, though, syntactic analysis succeeds only when sentences are well-formed, are constructed in a reasonably unambiguous fashion, with no sly, ironic or sarcastic intent on the part of the speaker. Outside of this domain, straightforward syntactic methods are doomed. Perhaps some simpler approaches can provide a more robust understanding of slang, euphemisms or even of "ungrammatical" or sloppy dialects. More difficult are situations that require the semantic backdrop of the conversation, knowledge of the speaker's role and intent, grasp of the social posturing and social dynamics of the text. Such dynamics can radically alter meaning, and strongly decouple the semantics from the syntax – the superficial "common-sense" interpretation obtained from the syntactic structure of a sentence may no longer apply (but curiously, would still be required to fully appreciate a witty twist of phrase).

The rules described in this note were linguist-developed, painstakingly crafted by hand to capture semantic content. The authors are involved in active research to automatically learn such rules, ideally using a form of unsupervised learning, such as that described in Dekang Lin's DIRT (Lin and Pantel, 2001) or Poon & Domingos USP (Domingos et al., 2006).

The source code for RelEx is publicly available, under the Apache License; it is written in Java. It depends on Link Grammar, which is publicly available under the BSD license, the Link Grammar parser is written in C. Both projects are actively maintained, and are kept in a stable or "production quality" state.

References

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *CrossParser '08: Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Pedro Domingos, Stanley Kok, Hoifung Poon, Matthew Richardson, and Parag Singla. 2006. Unifying logical and statistical ai. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, pages 2–7. AAAI Press.

Ben Goertzel. 2009. Opencogprime: A cognitive synergy based architecture for embodied artificial general intelligence. In *Proceedings of ICCI*, Hong Kong.

David Hart and Ben Goertzel. 2008. Opencog: A software framework for integrative artificial general intelligence. In *Proceedings of the First Conference on Artificial General Intelligence*. IOS Press.

Dekang Lin and Patrick Pantel. 2001. Dirt: Discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 323–328. ACM Press.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proc. Workshop on the Evaluation of Parsing Systems*, Granada.

Bill MacCartney Marie-Catherine de Marneffe and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC-06)*, pages 449–454.

Igor A. Mel'cuk and Alain Polguere. 1987. A formal lexicon in meaning-text theory. *Computational Linguistics*, 13:261–275.

Mike Ross, Linas Vepstas, and Ben Goertzel. 2005. Relex semantic relationship extractor. http://opencog.org/wiki/RelEx.

Daniel Sleator and Davy Temperley. 1991. Parsing english with a link grammar. Technical report, Carnegie Mellon University Computer Science technical report CMU-CS-91-196.

Daniel D. Sleator and Davy Temperley. 1993. Parsing english with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292.

James Steele, editor. 1990. *Meaning-Text Theory: Linguistics, Lexicography, and Implications*. University of Ottowa Press.