

GETARUNS: A HYBRID SYSTEM FOR SUMMARIZATION AND QUESTION ANSWERING

Rodolfo Delmonte

Ca' Garzoni-Moro, San Marco 3417

Università "Ca Foscari"

30124 - VENEZIA

Tel. 39-041-2579464/52/19 - Fax. 39-041-2577910

E-mail: delmont@unive.it - website: project.cgm.unive.it

Abstract

Information Retrieval, Summarization and Question Answering need accurate linguistic information with a much higher coverage than what is being recently offered by currently available parsers, so we assume that the starting point of any interesting application in those fields must necessarily be a good syntactic-semantic parser. The system presented in the paper has undergone extensive testing and the parser has been trained on available test suites and the Remedia corpus texts, one of which will be commented in some detail.

1. Introduction

In this paper we present an approach to natural language processing which we define as "hybrid", in which symbolic and statistical approaches are reconciled. In particular, the search space for syntactic analysis is inherently deterministic, in that it is severely limited by the grammar of the specific language, which in turn is constituted by a set of peripheral rules to be applied in concomitance with the more general rules of core grammar. Variations on these rules are only determined by genre, which can thus contribute a new set of peripheral rules, or sometimes just a set of partially overlapping rules to the ones already accepted by a given linguistic community. As far as parsing is concerned, we purport the view that the implementation of sound parsing algorithm must go hand in hand with sound grammar construction. Extragrammaticalities can be better coped with within a solid linguistic framework rather than without it. Our parser is a rule-based deterministic parser in the sense that it uses a

lookahead and a Well-Formed Substring Table to reduce backtracking (Delmonte, 2000a). It also implements Finite State Automata in the task of tag disambiguation (Delmonte, 2000b), and produces multiwords whenever lexical information allows it. In our parser (Delmonte, 2000c), we use a number of parsing strategies and graceful recovery procedures which follow a strictly parameterized approach to their definition and implementation. Recovery procedures are also used to cope with elliptical structures and uncommon orthographic and punctuation patterns. A shallow or partial parser, in the sense of Abney(1996), is also implemented and always activated before the complete parse takes place, in order to produce the default baseline output to be used by further computation in case of total failure. In that case partial semantic mapping will take place where no Logical Form is being built and only referring expressions are asserted in the Discourse Model – but see below.

1.1 Intelligent Sentence Extraction

The use of NLP techniques in IR/IE is in our opinion mandatory as a preliminary step towards the summarization itself: Intelligent Sentence Extraction (hence ISE) is the first step to produce a summary which in our case is strongly based on the use of NLP techniques. Its main features are: full tokenization, FSA restricted multiword creation, POS tagging limited to content words. The aim of ISE is to produce an 800 words extract to be used by GETARUNS, the system for summarization proper.

In this preliminary phase we rely on statistical techniques for extracting keywords to be used as topics. However before Sentence Extraction takes place, we perform a search for Topic Density which for each most frequent wordform computes a lemma and a tag. This allows to go

beyond stemming by taking into account only noun/verbs and denominal adjectives. The final Sentence Extraction step computes Dangling Sentences, those depending on some previous discourse element and tries to amend that by adding the previous sentence to the final output.

2. General Parser and Anaphora Resolution Module

GETARUNS, the system for text understanding developed at the University of Venice, is equipped with three main modules: a lower module for parsing where sentence strategies are implemented; a middle module for semantic interpretation and discourse model construction which is cast into Situation Semantics; and a higher module where reasoning and generation takes place (Delmonte, 2000c).

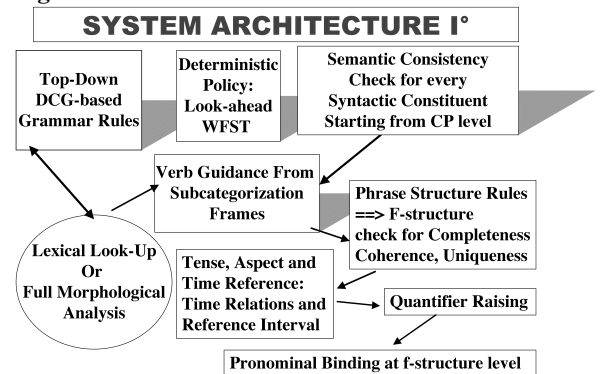
The system is based on LFG theoretical framework (see Bresnan, J. 2001) and has a highly interconnected modular structure. It is a top-down depth-first DCG-based parser written in Prolog which uses a strong deterministic policy by means of a lookahead mechanism with a WFST to help recovery when failure is unavoidable due to strong attachment ambiguity.

It is divided up into a pipeline of sequential but independent modules which realize the subdivision of a parsing scheme as proposed in LFG theory where a c-structure is built before the f-structure can be projected by unification into a DAG (Delmonte, 2002).

Syntactic and semantic information is accessed and used as soon as possible: in particular, both categorial and subcategorization information attached to predicates in the lexicon is extracted as soon as the main predicate is processed, be it adjective, noun or verb, and is used to subsequently restrict the number of possible structures to be built (see Fig.1 below).

The output of grammatical modules is fed then onto the Binding Module(BM) which activates an algorithm for anaphoric binding. Antecedents for pronouns are ranked according to grammatical function, semantic role, inherent features and their position at f-structure. Eventually, this information is added into the original f-structure graph and then passed on to the Discourse Module(DM).

Fig.1 GETARUNS' Sentence Level Modules



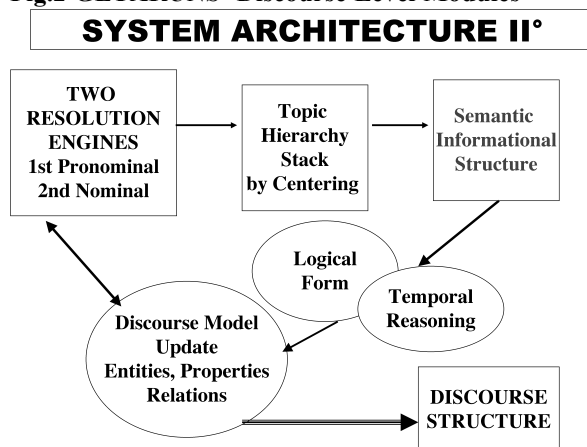
The grammar is equipped with a core lexicon containing most frequent 5000 fully specified inflected word forms where each entry is followed by its lemma and a list of morphological features, organised in the form of attribute-value pairs. However, morphological analysers for English are also available with big root dictionaries (25,000 for English) which only provide for syntactic subcategorization, though. In addition to that there are all lexical forms provided by a fully revised version of COMLEX, and in order to take into account phrasal and adverbial verbal compound forms, we also use lexical entries made available by UPenn and TAG encoding. Their grammatical verbal syntactic codes have then been adapted to our formalism and are used to generate a subcategorization schemes with an aspectual and semantic class associated to it – however no restrictions can reasonably be formulated on arguments of predicates. Semantic inherent features for Out of Vocabulary Words, be they nouns, verbs, adjectives or adverbs, are provided by a fully revised version of WordNet - plus EuroWordnet, with a number of additions coming from computer, economics, and advertising semantic fields - in which we used 75 semantic classes similar to those provided by CoreLex.

2.1 The Upper Module

GETARUNS, has a highly sophisticated linguistically based semantic module which is used to build up the Discourse Model. Semantic processing is strongly modularized and distributed amongst a number of different submodules which take care of Spatio-Temporal

Reasoning, Discourse Level Anaphora Resolution, and other subsidiary processes like Topic Hierarchy which cooperate to find the most probable antecedent of coreferring and cospecifying referential expressions when creating semantic individuals. These are then asserted in the Discourse Model (hence the DM), which is then the sole knowledge representation used to solve nominal coreference. The system uses two resolution submodules which work in sequence: they constitute independent modules and allow no backtracking. The first one is fired whenever a free sentence external pronoun is spotted; the second one takes the results of the first submodule and checks for nominal anaphora. They have access to all data structures contemporarily and pass the resolved pair, anaphor-antecedent to the following modules. See Fig.2 below. Semantic Mapping is performed in two steps: at first a Logical Form is produced which is a structural mapping from DAGs onto of unscoped well-formed formulas. These are then turned into situational semantics informational units, infons which may become facts or sits. Each unit has a relation, a list of arguments which in our case receive their semantic roles from lower processing – a polarity, a temporal and a spatial location index.

Fig.2 GETARUNS' Discourse Level Modules



2.1 Building the Discourse Model

All entities and their properties are asserted in the DM with the relations in which they are involved; in turn the relations may have modifiers - sentence level adjuncts and entities may also have modifiers or attributes. Each entity

has a polarity and a couple of spatiotemporal indices which are linked to main temporal and spatial locations if any exists; else they are linked to presumed time reference derived from tense and aspect computation. Entities are mapped into semantic individual with the following ontology: on first occurrence of a referring expression it is asserted as an INDividual if it is a definite or indefinite expression; it is asserted as a CLASS if it is quantified (depending on quantifier type) or has no determiner. Special individuals are ENTs which are associated to discourse level anaphora which bind relations and their arguments. Finally, we have LOCs for main locations, both spatial and temporal. If it has a cardinality determined by a number, it is plural or it is quantified (depending on quantifier type) it is asserted as a SET and the cardinality is simply inferred in case of naked plural, i.e. in case of collective nominal expression it is set to 100, otherwise to 5. On second occurrence of the same nominal head the semantic index is recovered from the history list and the system checks whether it is the same referring expression:

- in case it is definite or indefinite with a predicative role and no attributes nor modifiers nothing is done;
- in case it has different number - singular and the one present in the DM is a set or a class nothing happens;
- in case it has attributes and modifiers which are different and the one present in the DM has none, nothing happens;
- in case it is quantified expression and has no cardinality, and the one present in the DM is a set or a class, again nothing happens.

In all other cases a new entity is asserted in the DM which however is also computed as being included in (a superset of) or by (a subset of) the previous entity.

2.2 The Partial System

The system switches automatically to a Partial or Shallow modality in case of failure in the Complete modality. The Partial system has standard components. like a statistical/syntactic tagger and a cascaded shallow parser which in a final run turns syntactic constituents into functionally labelled arguments/adjuncts. The "Partial" modality when activated allows the

system to relax both parsing and semantic mapping constraints thus producing a "partial" semantic mapping. Here partial refers both to the possibility that the parser output a parse which covers only parts of the input sentence, leaving out some linguistic material unparsed. It may also refer to cases in which the predicate-argument mapping does not succeed due to lack of adequate subcategorization information. Finally it may refer to the cases of missed anaphora resolution, in which the actual semantic mapping does not simply make the right inferences due to the presence of bridging expressions which are mismatched with their coreferring expression, or else for the lack of the appropriate inferential information to fire the inference in the first place. In the "Partial" modality, the list of Essential Properties of Linguistic Objects which can be derived is as follows:

1. Grammatical properties of linguistic objects
 - i. Functional Features of NPs
 - ii. Selectional Restrictions of NPs #
 - iii. Grammatical Functions of NPs
 - iv. Semantic Roles of all Phrases
2. Topic Hierarchies for referring expressions
3. Discourse Model of facts and locations
4. External World Knowledge Base

where we indicate with a grid Selectional restrictions of NPs that will only be used to induce the right adjunct/oblique semantic role in presence of a given preposition. In addition, there is no attempt at producing a full semantic interpretation, no logical form is being generated due to the uncertainty at clause structure building.

2.3 Getaruns at work

We will show how Getaruns computes the DM by presenting the output of the system for the «Maple Syrup» text made available by Mitre for the ANLP2000 Workshop(see Hirschman et al.). Here below is the original text which is followed by the DM with the Semantic Database of Entities and Relations of the Text World.

How Maple Syrup is Made

Maple syrup comes from sugar maple trees. At one time, maple syrup was used to make sugar. This is why the tree is called a "sugar" maple tree.

Sugar maple trees make sap. Farmers collect the sap. The best time to collect sap is in February and March. The nights must be cold and the days warm.

The farmer drills a few small holes in each tree. He puts a spout in each hole. Then he hangs a bucket on the end of each spout. The bucket has a cover to keep rain and snow out. The sap drips into the bucket. About 10 gallons of sap come from each hole.

1. Who collects maple sap? (Farmers)
2. What does the farmer hang from a spout? (A bucket)
3. When is sap collected? (February and March)
4. Where does the maple sap come from? (Sugar maple trees)
5. Why is the bucket covered? (to keep rain and snow out)

2.4 Discourse Model for the text organized sentence by sentence

We list here below the DM related to the most relevant sentences addressed by the QA module:

1.How Maple Syrup is Made

```
loc(infon1, id1, [arg:main_tloc, arg:tr(f2_es1)])
class(infon2, id2)
fact(infon3, Maple, [ind:id2], 1, id1, univ)
fact(infon4, inst_of, [ind:id2, class:edible_substance], 1, univ, univ)
fact(infon5, isa, [ind:id2, class:Syrup], 1, id1, univ)
ind(infon6, id3)
fact(infon7, inst_of, [ind:id3, class:plant_life], 1, univ, univ)
fact(infon8, isa, [ind:id3, class:Maple], 1, id1, univ)
in(infon9, id3, id2)
fact(id5, make, [agent:id2, theme_aff:id4], 1, tes(f2_es1), univ)
fact(infon13, isa, [arg:id5, arg:ev], 1, tes(f2_es1), univ)
fact(infon14, isa, [arg:id6, arg:tloc], 1, tes(f2_es1), univ)
fact(infon15, plu_perf, [arg:id6], 1, tes(f2_es1), univ)
fact(infon16, time, [arg:id5, arg:id6], 1, tes(f2_es1), univ)
fact(infon17, how, [arg:id5], 1, tes(f2_es1), univ)
before(tes(f2_es1), tes(f2_es1))
includes(tr(f2_es1), id1)
```

2.Maple syrup comes from sugar maple trees

```
loc(infon20, id7, [arg:main_sloc, arg:tree])
in(infon21, id8, id7)
set(infon22, id8)
card(infon23, id8, 5)
fact(infon24, sugar_maple, [ind:id8], 1, id1, id7)
fact(infon25, inst_of, [ind:id8, class:plant_life], 1, univ, univ)
fact(infon26, isa, [ind:id8, class:tree], 1, id1, id7)
ind(infon27, id9)
fact(infon28, inst_of, [ind:id9, class:plant_life], 1, univ, univ)
fact(infon29, isa, [ind:id9, class:sugar_maple], 1, univ, univ)
in(infon30, id9, id3)
class(infon32, id10)
fact(infon33, isa, [ind:id10, class:sugar], 1, univ, univ)
fact(infon34, of, [arg:id10, specif:id9], 1, univ, univ)
fact(id11, come, [actor:id2, locat:id8], 1, tes(f1_es2), id7)
fact(infon38, isa, [arg:id11, arg:ev], 1, tes(f1_es2), id7)
fact(infon39, isa, [arg:id12, arg:tloc], 1, tes(f1_es2), id7)
fact(infon40, pres, [arg:id12], 1, tes(f1_es2), id7)
```

fact(infon41, time, [arg:id11, arg:id12], 1, tes(fl_es2), id7)
during(tes(fl_es2), tes(f2_es1))
includes(tr(fl_es2), univ)

3. At one time, maple syrup was used to make sugar
class(infon42, id13)
fact(infon43, inst_of, [ind:id13, class:substance], 1, univ, univ)
fact(infon44, isa, [ind:id13, class:sugar], 1, univ, id7)
in(infon45, id13, id9)
fact(id14, make, [agent:id2, theme_aff:id13], 1, tes(finfl_es3), id7)
fact(infon47, isa, [arg:id14, arg:ev], 1, tes(finfl_es3), id7)
fact(infon48, isa, [arg:id15, arg:tloc], 1, tes(finfl_es3), id7)
fact(infon49, pres, [arg:id15], 1, tes(finfl_es3), id7)
fact(infon50, time, [arg:id14, arg:id15], 1, tes(finfl_es3), id7)
fact(id16, use, [theme_aff:id2, result:id14], 1, tes(f2_es3), id7)
fact(infon51, isa, [arg:id16, arg:pr], 1, tes(f2_es3), id7)
fact(infon52, isa, [arg:id17, arg:tloc], 1, tes(f2_es3), id7)
fact(infon53, plu_perf, [arg:id17], 1, tes(f2_es3), id7)
fact(infon54, time, [arg:id16, arg:id17], 1, tes(f2_es3), id7)
fact(infon55, at_one_time, [arg:id16], 1, tes(f2_es3), id7)
before(tes(f2_es3), tes(f2_es1))
includes(tr(f2_es3), univ)

4. This is why the tree is called a "sugar" maple tree
ent(infon61, id18)
fact(infon62, prop, [arg:id18, disc_set:[id16:use:[theme_aff:id2, result:id14]], 1, univ, id7)
ind(infon63, id19)
fact(infon64, tree, [nil:id19], 1, univ, id7)
fact(infon65, inst_of, [ind:id19, class:plant_life], 1, univ, univ)
fact(infon66, isa, [ind:id19, class:tree], 1, univ, id7)
in(infon67, id19, id8)
ind(infon68, id20)
fact(infon69, inst_of, [ind:id20, class:thing], 1, univ, univ)
fact(infon70, isa, [ind:id20, class:reason], 1, univ, id7)
fact(infon72, reason, [nil:id18], 1, univ, id7)
fact(id21, be, [prop:infon72], 1, tes(f15_es4), id7)
fact(id22, call, [actor:id19, theme:id10, prop:infon72], 1, tes(f15_es4), id7)
fact(infon73, isa, [arg:id22, arg:st], 1, tes(f15_es4), id7)
fact(infon74, isa, [arg:id23, arg:tloc], 1, tes(f15_es4), id7)
fact(infon75, pres, [arg:id23], 1, tes(f15_es4), id7)
fact(infon82, time, [arg:id22, arg:id23], 1, tes(f5_es4), id7)
during(tes(f15_es4), tes(f2_es3))
includes(tr(f15_es4), univ)

5. Sugar maple trees make sap
class(infon85, id24)
fact(infon86, inst_of, [ind:id24, class:substance], 1, univ, univ)
fact(infon87, isa, [ind:id24, class:sap], 1, univ, id7)
fact(id26, make, [agent:id8, theme_aff:id24], 1, tes(fl_es5), id7)
fact(infon92, isa, [arg:id26, arg:ev], 1, tes(fl_es5), id7)
fact(infon93, isa, [arg:id27, arg:tloc], 1, tes(fl_es5), id7)
fact(infon94, pres, [arg:id27], 1, tes(fl_es5), id7)
fact(infon98, time, [arg:id26, arg:id27], 1, tes(fl_es5), id7)
during(tes(fl_es5), tes(f15_es4))
includes(tr(fl_es5), univ)

6. Farmers collect the sap
set(infon99, id28)
card(infon100, id28, 5)
fact(infon101, inst_of, [ind:id28, class:man], 1, univ, univ)
fact(infon102, isa, [ind:id28, class:farmer], 1, univ, id7)
fact(id29, collect, [agent:id28, theme_aff:id24], 1, tes(fl_es6), id7)
fact(infon105, isa, [arg:id29, arg:ev], 1, tes(fl_es6), id7)
fact(infon106, isa, [arg:id30, arg:tloc], 1, tes(fl_es6), id7)
fact(infon107, pres, [arg:id30], 1, tes(fl_es6), id7)
fact(infon108, time, [arg:id29, arg:id30], 1, tes(fl_es6), id7)

during(tes(fl_es6), tes(fl_es5))
includes(tr(fl_es6), univ)

7. The best time to collect sap is in February and March
ind(infon110, id32)
fact(infon111, best, [ind:id32], 1, univ, id7)
fact(infon112, inst_of, [ind:id32, class:time], 1, univ, univ)
fact(infon113, isa, [ind:id32, class:time], 1, univ, id7)
set(infon114, id33)
card(infon115, 2)
fact(infon116, inst_of, [ind:id33, class:time], 1, univ, univ)
fact(infon117, isa, [ind:id33, class:[march, February]], 1, univ, id7)
fact(id35, collect, [agent:id28, theme_aff:id24], 1, tes(finfl_es7), id7)
fact(infon118, isa, [arg:id35, arg:ev], 1, tes(finfl_es7), id7)
fact(infon119, isa, [arg:id36, arg:tloc], 1, tes(finfl_es7), id7)
fact(infon120, nil, [arg:id36], 1, tes(finfl_es7), id7)
fact(infon121, [march, February], [arg:id32], 1, univ, id7)
fact(id37, be, [prop:id35, prop:infon130], 1, tes(fl_es7), id7)
fact(infon122, isa, [arg:id37, arg:st], 1, tes(fl_es7), id7)
fact(infon123, isa, [arg:id38, arg:tloc], 1, tes(fl_es7), id7)
fact(infon124, pres, [arg:id38], 1, tes(fl_es7), id7)
fact(infon125, time, [arg:id37, arg:id38], 1, tes(fl_es6), id7)
during(tes(fl_es7), tes(fl_es6))
includes(tr(fl_es7), univ)

.....
9. The farmer drills a few small holes in each tree
class(infon163, id38)
fact(infon164, small, [ind:id38], 1, univ, id7)
fact(infon165, inst_of, [ind:id38, class:place], 1, univ, univ)
fact(infon166, isa, [ind:id38, class:holes], 1, univ, univ)
fact(id47, drill, [agent:id28, theme_aff:id38], 1, tes(fl_es9), id7)
fact(infon170, isa, [arg:id47, arg:ev], 1, tes(fl_es9), id7)
fact(infon171, isa, [arg:id48, arg:tloc], 1, tes(fl_es9), id7)
fact(infon172, pres, [arg:id48], 1, tes(fl_es9), id7)
fact(infon173, in, [arg:id45, locat:id19], 1, tes(fl_es9), id7)
fact(infon157, time, [arg:id47, arg:id48], 1, tes(fl_es9), id7)
during(tes(fl_es9), tes(fl_es8))
includes(tr(fl_es9), univ)

.....
12. The bucket has a cover to keep rain and snow out
class(infon218, id59)
fact(infon219, inst_of, [ind:id59, class:thing], 1, univ, univ)
fact(infon220, isa, [ind:id59, class:cover], 1, id53, id7)
fact(infon222, cover, [nil:id54], 1, id53, id7)
fact(id60, have, [actor:id54, prop:infon222, prop:id65], 1, tes(fl_es12), id7)
fact(infon223, isa, [arg:id60, arg:st], 1, tes(fl_es12), id7)
fact(infon224, isa, [arg:id61, arg:tloc], 1, tes(fl_es12), id7)
fact(infon225, pres, [arg:id61], 1, tes(fl_es12), id7)
fact(infon227, isa, [arg:id62, arg:rain], 1, tes(fl_es12), id7)
fact(infon228, isa, [arg:id63, arg:snow], 1, tes(fl_es12), id7)
fact(id65, keep_out, [agent:id54, theme_aff:id64], 1, tes(finfl_es12), id7)
fact(infon229, isa, [arg:id65, arg:pr], 1, tes(finfl_es12), id7)
fact(infon230, isa, [arg:id66, arg:tloc], 1, tes(finfl_es12), id7)
fact(infon231, pres, [arg:id66], 1, tes(finfl_es12), id7)
fact(infon232, time, [arg:id65, arg:id66], 1, tes(fl_es12), id7)
fact(infon233, coincide, [arg:id60, prop:id65], 1, tes(fl_es12), id7)
during(tes(fl_es12), tes(fl_es11))
includes(tr(fl_es12), id53)

...

14. About 10 gallons of sap come from each hole
 set(infon259, id69)
 card(infon260, id69, 10)
 fact(infon261, inst_of, [ind:id69, class:nquant], 1, univ, univ)
 fact(infon262, isa, [ind:id69, class:gallon], 1, id1, univ)
 fact(infon263, of, [arg:id69, specif:id24], 1, id1, univ)
 fact(id70, come, [agent:id69, theme_aff:id38], 1, tes(fl_es14), id7)
 fact(infon264, isa, [arg:id70, arg:ev], 1, tes(fl_es14), id7)
 fact(infon265, isa, [arg:id71, arg:tloc], 1, tes(fl_es14), id7)
 fact(infon266, pres, [arg:id71], 1, tes(fl_es14), id7)
 fact(infon267, time, [arg:id70, arg:id71], 1, tes(fl_es14), univ)
 during(tes(fl_es14), tes(fl_es13))
 includes(tr(fl_es14), id53)

3. Question-Answering

Coming now to Question Answering, the system accesses the DM looking for relations at first then for entities : entities are searched according to the form of the focussed element in the User DataBase of Question-Facts as shown below with the QDM for the first question:

User Question-Facts Discourse Model

q_loc(infon3, id1, [arg:main_tloc, arg:tr(fl_free_a)])
 q_ent(infon4, id2)
 q_fact(infon5, isa, [ind:id2, class:who], 1, id1, univ)
 q_fact(infon6, inst_of, [ind:id2, class:man], 1, univ, univ)
 q_class(infon7, id3)
 q_fact(infon8, inst_of, [ind:id3, class:coll], 1, univ, univ)
 q_fact(infon9, isa, [ind:id3, class:sap], 1, id1, univ)
 q_fact(infon10, focus, [arg:id2], 1, id1, univ)
 q_fact(id4, collect, [agent:id2, theme_aff:id3], 1, tes(fl_free_a), univ)
 q_fact(infon13, isa, [arg:id4, arg:pr], 1, tes(fl_free_a), univ)
 q_fact(infon14, isa, [arg:id5, arg:tloc], 1, tes(fl_free_a), univ)
 q_fact(infon15, pres, [arg:id5], 1, tes(fl_free_a), univ)

As to the current text, it replies correctly to the all questions. As to question 4, at first the system takes « come from » to be answered exhaustively by sentence 14 ; however, seen that «hole » is not computed with a « location » semantic role, it searches the DM for a better answer which is the relation linguistically expressed in sentence 9, where «holes » are drilled «in each tree». The « tree » is the Main Location of the whole story and « hole » in sentence 9 is inferentially linked to « hole » in sentence 14, by a chain of inferential inclusions. In fact, come_from does not figure in WordNet even though it does in our dictionary of synonyms. As to the fifth question, the system replies correctly.

1. Who collects maple sap? (Farmers)
2. What does the farmer hang from a spout?
(A bucket)

3. When is sap collected? (February and March)
4. Where does the maple sap come from?
(Sugar maple trees)
5. Why is the bucket covered? (to keep rain and snow out)

Another possible « Why » question could have been the following : « why the tree is called a "sugar" maple tree », which would have received the appropriate answer seen that the corresponding sentence has received an appropriate grammatical and semantic analysis. In particular, the discourse deictic pronoun « This » has been bound to the previous main relation « use » and its arguments so that they can be used to answer the « Why » question appropriately.

There is not enough space here to comment in detail the parse and the semantics (but see Delmonte 2000c); however, as far as anaphora resolution is concerned, the Higher Module computes the appropriate antecedent for the big Pro of the arbitrary SUBJECT of the infinitive in sentence n. 7, where the collecting action would have been left without an agent. This resolution of anaphora is triggered by the parser decision to treat the big Pro as an arbitrary pronominal and this information is stored at lexical level in the subcategorization frame for the name « time ».

Differently from what Schwitter et al. conclude in their paper, good linguistic processing can be achieved as long as strongly linguistically-based processing is performed. In the analysis they make of the semantic processing necessary in their opinion to account for the complexity of the task they frequently refer to the use of abductive reasoning: this is what is done in our system in order to cope with uncertain and/or insufficient information. For instance, with question n.4 the text only makes available information related to « maple syrup ». Since we start looking for relation, and the «come from » relation has a different linguistic description as SUBJECT argument, what we do is to try and see whether there is some inferential link between « sap » and « syrup ». This is not the case seen that WordNet does not link the two concepts explicitly. However both are classified as « substance » thus allowing the required inference to be fired – both are also taken as synonyms in our dictionary. The final question does not constitute a problem seen that the predicate «cover» has become a semantic

relation and is no longer a noun or a verb. Also worth noting is the fact that the question is not a real passive, but a quasi-passive or an ergative construction, so no agent should be searched for. So our conclusion is that the heart of a Q/A system should be a strongly restrictive pipeline of linguistically based modules which alone can ensure the adequate information for the knowledge representation and the reasoning processes required to answer natural language queries.

3.1.1 Evaluating GETARUNS approach to QA

Totally shallow approaches when compared to ours will always be lacking sufficient information for semantic processing at propositional level: in other words, as happens with our “Partial” modality, there will be no possibility of checking for precision in producing predicate-argument structures

Most systems would use some Word Matching algorithm that counts the number of words that appear in both the question and the sentence being considered after stripping them of stopwords: usually two words will match if they share the same morphological root after some stemming has taken place. Most QA systems presented in the literature rely on the classification of words into two classes: function and content words. They don't make use of a Discourse Model where input text has been transformed via a rigorous semantic mapping algorithm: they rather access tagged input text in order to sort best match words, phrases or sentences according to some matching scoring function.

It is also common knowledge the fact that only by introducing or increasing the amount of linguistic knowledge over crude IR-based systems will contribute substantial improvements. In particular, systems based on simple Named-Entity identification tasks are too rigid to be able to match phrase relations constraints often involved in a natural language query.

First objection is the impossibility to take into account pronominal expressions, their relations and properties as belonging to the antecedent, if no head transformation has taken place during the analysis process.

Second objection is the use of grammatical function labels, like SUBJ/OBJects without an evaluation of their relevance in the utterance structure: higher level or main clause SUBJ/OBJects are more important than other SUBJects. In addition, there is no attempt at semantic role assignment which would come from a basic syntactic/semantic tagging of governing verbs: a distinction into movement verbs, communication verbs, copulative verbs, psychic verbs etc. would suffice to assign semantic roles to main arguments if present.

It is usually the case that QA systems divide the question to be answered into two parts: the Question Target represented by the wh- word and the rest of the sentence; otherwise the words making up the yes/no question and then a match takes place in order to identify most likely answers in relation to the rest/whole of the sentence except for stopwords.

However, it is just the semantic relations that need to be captured and not just the words making up the question that matter. Some system implemented more sophisticated methods (notably Hovy et al.; Litkowski): syntactic-semantic question analysis. This involves a robust syntactic-semantic parser to analyse the question and candidate answers, and a matcher that combines word- and parse-tree-level information to identify answer passages more precisely.

3.1.2 Answering Generic Question

An important issue in QA is answering generic questions on the “aboutness” of the text, questions which may be answered by producing appropriate headlines or just a title. In our system, given the concomitant work of anaphora resolution modules and the semantic mapping into predicate-argument structures, this can be made as follows. The system collapses all entities and their properties, relations and attributes at the end of the analysis, by collecting them for ontological type; each semantic id receives a score for topichood thus allowing a ranking of the entities; in addition, starred facts are inherited by the inclusion relation specified by the “in” semantic predicate, as in the case of the “specifying” relation between “sugar” and “maple”. Here below we list the most relevant entities for the text reported above:

```
entity(class,id2,115,facts([
fact(infon3, 'Maple', [ind:id2], 1, T, P),
fact(infon4, inst_of, [ind:id2, class:edible_animal], 1, T, P),
fact(infon5, isa, [ind:id2, class:'Syrup'], 1, T, P),
fact(id5, make, [theme_bound:id2, agent:id4], 1, T, P),
fact(id11, come, [actor:id2, locat:id8], 1, T, P),
fact(id14, make, [agent:id2, theme_aff:id13], 1, T, P),
fact(id16, use, [theme_aff:id2, result:id14], 1, T, P)))).
```

```
entity(class,id30,77,facts([
fact(infon114, inst_of, [ind:id30, class:man], 1, T, P),
fact(infon115, isa, [ind:id30, class:farmer], 1, T, P),
fact(id39, drill, [agent:id30, theme_aff:id38], 1, T, P),
fact(id42, put, [agent:id30, theme_aff:id41, locat:id38], 1, T, P),
fact(id48, hang, [agent:id30, theme_aff:id44], 1, T, P)))).
```

In this way, an appropriate answer to the question "What is the text about" can be generated directly from the entity list by picking up relations and properties of the most relevant individuals, sets and classes (see Delmonte 2000c).

4. System Evaluation

The complete system has been built for a restricted linguistic domain and evaluated on the basis of the texts making up the domain: 3500 total words, 265 sentences. The performance is 95% correct. The system has been tested with a set of texts derived from newspapers, narrative texts, children stories summing up to 10,000 words where we got the same results: However, updating and tuning of the system is required for each new text whenever a new semantic relation is introduced by the parser and the semantic does not provide the appropriate mapping. For instance, consider the case of the constituent "holes in the tree", where the syntax produces the appropriate structure but the semantics does not map "holes" as being in a LOCATION semantic relation with "tree". In lack of such a semantic role information a dummy "MODal" will be produced which however will not generate the adequate semantic mapping in the DM and the meaning is lost.

As to the partial system, it has been used for DUC summarization contest, i.e. it has run over approximately 1 million words, including training and test sets, for a number of sentences totalling over 50K. We tested the "Partial" modality with an additional 90,000 words texts taken from the testset made available by DUC 2002 contest. On a preliminary perusal of the results, we have calculated a 85% Precision on parsing and an 80% on semantic mapping.

However evaluating such results requires a manually annotated database in which all linguistic properties have been carefully decided by human annotators. In lack of such a database, we are unable to provide precise performance data.

References

- Abney, S. 1996. Part-of-Speech Tagging and Partial Parsing, in Ken Church, Steve Young, and Gerrit Bloothoof, eds. *Corpus-Based Methods in Language and Speech*, Kluwer Academic Publishers, Dordrecht.
- Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Blackwells.
- Brill, E., Lin, J., Banko, M., Dumais, S., & Ng, A. (2002). Data-Intensive Question Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. 122-131.
- Delmonte R., 2000a. Parsing Preferences and Linguistic Strategies, in *LDV-Forum - Zeitschrift fuer Computerlinguistik und Sprachtechnologie - "Communicating Agents"*, Band 17, 1,2, pp. 56-73.
- Delmonte R., 2000b. Shallow Parsing And Functional Structure In Italian Corpora, *Proc. LREC*, Athens, pp.113-119.
- Delmonte R. 2000c. Generating from a Discourse Model, *Proc. MT-2000*, BCS, Exeter, pp.25-1/10.
- Delmonte R. 2002. GETARUN PARSER - A parser equipped with Quantifier Raising and Anaphoric Binding based on LFG, *Proc. LFG2002 Conference*, Athens, pp.130-153, at <http://csli-publications.stanford.edu/hand/miscpubsonline.html>.
- Hirschman, L. Marc Light, Eric Breck, & J. D. Bugar. 1999. Deep Read: A reading comprehension system. In *Proc. A CL '99*. University of Maryland.
- Schwiter R., D. Mollà, R. Fournier & M. Hess, 2000. Answer Extraction: Towards better Evaluations of NLP Systems. In *Proc. Works. Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*, Seattle, 20-27.
- Hovy, E., U. Hermjakob, & C. Lin. (2002a). The Use of External Knowledge in Factoid QA. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. 644-652.
- Litkowski, K. C. (2001). Syntactic Clues and Lexical Resources in Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Ninth Text Retrieval Conference (TREC-9)*. 157-166.
- Ravichandran, D. & E. Hovy. (2002). Learning Surface Text Patterns for a Question Answering System. Proceedings of the 40th ACL. Philadelphia, PA., 41-7.