A Weighted Robust Parsing Approach to Semantic Annotation

Hatem Ghorbel and Vincenzo Pallotta
LITH-MEDIA group
Swiss Federal Institute of Technology
IN Ecublens, 1015 Lausanne, Switzerland
{ghorbel,pallotta} @di.epfl.ch

Abstract

This paper proposes a grammar-based approach to semantic annotation which combines the notions of robust parsing and fuzzy grammars. We present an overview of a preliminary research aimed to generalize some results from a recent project on interaction through speech with information systems where techniques based on the above notions have been successfully applied. The goal of the article is to give a development environment to linguists.

1 Introduction

In this article we are mainly interested in semantic annotation (Sperberg-McQueen and Burnard, 1994). We are considering the Information Extraction (I.E.) problem as a semantic annotation problem: extracting information is finding the relevant terms that contribute to describe an appropriate semantic structure of the text. Some of the most important works in I.E. have been dealing with domain dependent documents like (Moll et al., 1998; Hobbs et al., 1996). Both systems employ complex analysis schemas. Assigning semantic field tags is in general a difficult task. This is due at least to the crucial need of the domain knowledge and also of the linguistic knowledge. Our approach considers that for some specific domains a semantic annotation can be achieved by a *light parsing* of the text which is based on the user of certain *cue-words* as a heuristic for describing its semantic structure.

1.1 A case study in query generation

The availability of a large collection of annotated telephone calls for querying the Swiss phone-book database (i.e the Swiss French PolyPhone corpus) allowed us to experiment our recent findings in robust text analysis obtained in the context of the Swiss National Fund research project ROTA (Robust Text Analysis), and in the recent Swisscom funded project ISIS (Interaction through Speech with Information Systems)[1] (Chappelier et al., 1999). This database contains 4293 simulated recordings related to the "111" Swisscom service calls. For instance a query call like:

```
Bonjour j'aimerais un numéro de
   téléphone à Saignelegier c'est
   Mottaz m o deux ta z Monique rue du
   printemps numéro quatre
```

would produce the following query frame filling for the Swiss Phone-book database:

```
Nom de famille / Firme: MOTTAZ
Prénom / Autres informations: MONIQUE
Rue, numéro: rue du PRINTEMPS, 4
NPA, localité: SAIGNELEGIER.
```

The goal of semantic annotation is to provide a tree structure which can be superposed over the flat sentence. This structure can be supported by a PROLOG compound term consisting of "tag" functors and list arguments. Moreover this same structure can also be supported by the SGML structural representation. The translation between the two models is an intuitive task and an example of such translation is provided by the two following corresponding representations for a possible query call schema.

PROLOG:

```
s([...,announce([...]),
   query([...,
      name([...,
         fam_name([...]),...,
         first_name([...]),
        ...]),...,
      address([...,
         street([...]),...,
         city([...]),
          ...]),
     ...])
 ...]).
```

SGML:

```
...
<announce>
...
</announce>
<query>
   <name>
```

[1]The final report of ISIS project is available at http://lithwww.epfl.ch/~pallotta/isis.html

```
            ...
            <fam_name> ... </fam_name>
            ...
            <first_name> ... </first_name>
            ...
        </name>
        ...
        <address>
            ...
            <street> ... </street>
            ...
             <city> ... </city>
            ...
        </address>
        ...
    </query>
    ...
```

1.1.1 Processing phases

The processing of the corpus data is performed at various linguistic levels by modules organized into a pipeline. Each module assumes as input the output of the preceding module. The main goal of this architecture is to understand how far it is possible to go without using any kind of feedback and interactions among different linguistic modules. At a first stage, morphologic and syntactic processing[2] is applied to the output from the *speech recognizer* module which usually produces a huge word-graph hypothesis. Thus the forest of syntactic trees produced by this phase have been used to achieve two goals:

1. The n-best analyses are used to disambiguate speech recognizer hypotheses
2. They served as supplementary input for the robust semantic analysis that we performed, that had as goal the production of query frames for the information system.

Although robustness can be considered as being applied at either a syntactic or semantic level, we believe it is generally at the semantic level that it is most effective. This robust analysis needs a model of the domain in which the system operates, and a way of linking this model to the lexicon used by the other components. The degree of detail required of the domain model used by the robust analyzer depends upon the ultimate task that must be performed — in our case, furnishing a query to an information system. The results of the processing phase of the previous example is represented below as an SGML annotation:

```
<announce>
 Bonjour j'aimerais un
</announce>
<query> numéro de télêphone à Saignelegier
   <name>
       <fam_name> c'est Mottaz m o deux ta z
       </fam_name>
       <first_name> Monique </first_name>
   </name>
   <address>
       <street> rue du printemps </street>
       <number> numéro quatre </number>
       <city> </city>
   </address>
</query>
```

2 Methodology

In this section we propose the use of a "light-parser" for doing sentence-level semantic annotation. The main idea comes from the observation that annotation does not always need to rely on the deep structure of the sentence (e.g. at morpho-syntactic level). It is sometimes sufficient to find some *cue-words* which allow us to locate the logical sub-structures of the sentence. If the domain is simple enough, this task can be easily mechanized. A similar approach, using finite state parsing technology, has been proposed by Grefenstette in (Grefenstette, 1996) where the main applications are slanted to the extraction of syntactic information.

2.1 Robust Definite Clause Grammars

LHIP (Left-corner Head-driven Island Parser) (Ballim and Russell, 1994; Lieske and Ballim, 1998) is a system which performs robust analysis of its input, using a grammar defined in an extended form of the PROLOG Definite Clause Grammars (DCGs). The chief modifications to the standard PROLOG 'grammar rule' format are of two types: one or more right-hand side (RHS) items may be marked as 'heads' (e.g. using a leading '*'), and one or more RHS items may be marked as 'ignorable' (e.g. using a leading '-'). LHIP employs a different control strategy from that used by PROLOG DCGs, in order to allow it to cope with ungrammatical or unforeseen input. The behavior of LHIP can best be understood in terms of the complementary notions of **span** and **cover**. A grammar rule is said to produce an island which **spans** input terminals t_i to t_{i+n} if the island starts at the i^{th} terminal, and the $i + n^{th}$ terminal is the terminal immediately to the right of the last terminal of the island. A rule is said to **cover** m items if m terminals are consumed in the span of the rule.

[2]Our partner institution ISSCO (Institute Dalle Molle, University of Geneva) performed this analysis phase using tools that were developed in the European Linguistics Engineering project MULTEXT. For syntactic analysis, ISSCO developed a Feature Unification Grammar based on a small sample of the Polyphone data. This grammar was taken by another of our partners (the Laboratory for Artificial Intelligence of the Swiss Federal Institute of Technology, Lausanne) and converted into a probabilistic context-free grammar, which was then applied to a sample of 500 entries from the Polyphone data.

Thus $m \leq n$. If $m = n$ then the rule has completely covered the span. As implied here, rules need not cover all of the input in order to succeed.

2.1.1 Weighted LHIP rules

The main goal of introducing weights into LHIP rules is to induce a partial order over the generated hypotheses. The following schema illustrate how to build a simple weighted rule in a compositional fashion where the resulting weight is computed from the sub-constituents using the minimum operator. Weights are real numbers in the interval $[0, 1]$.

```
cat(cat(Hyp),Weight) ~~>
          sub_cat1(H1,W1),
            ...,
          sub_catn(Hn,Wn),
          {app_list([H1,...,Hn],Hyp),
           min_list([W1,...,Wn],Weight)}.
```

This strategy is not the only possible since the LHIP formalism allows a greater flexibility. Without entering into formal details we can observe that if we strictly follow the above schema and we impose a cover threshold of 1 we are dealing with *fuzzy DCG grammars* (Lee and Zadeh, 1969; Asveld, 1996). We actually extend this class of grammars with a notion of *fuzzy-robustness* where weights are used to compute confidence factors for the membership of islands to categories[3]. The order of constituents may play an important role in assigning weights for different rules having the same number and type of constituents. Each LHIP rule returns a weight together with a term which will contribute to build the resulting structure. The confidence factor for a pre-terminal rule has been assigned statically on the basis of the rule designer's domain knowledge.

2.2 The methodology at work

In our case study we try to integrate the above principles in order to effectively compute annotation hypotheses for the query generation task. This can be done by building a lattice of *annotation hypotheses* and possibly selecting the best one. This lattice is generated by means of a LHIP *weighted grammar* which is used to extract and assemble what we called *semantic constituents*. At the end of this process we presumably obtain suitable annotations from which we will able to extract the content of the query (e.g. name, address, city, etc.). The rules are designed taking into consideration the following kind of knowledge:

Domain Knowledge is exploited to provide quantitative support (or confidence factor) to our rules.

[3]Development of this notion is currently under investigation and not yet formalized.

Linguistic Knowledge (as for instance previous POS tagging or syntactic analysis) is used for determining constraints in order to prune the hypotheses space.

Lexical knowledge: As pointed out in (Basili and M.T., 1997), lexical knowledge plays an important role in Information Extraction since it can contribute in guiding the analysis process at various linguistic level. In our case we are concerned with lexical knowledge when we need to specify lexical LHIP rules which represent the building blocks of our parsing system. *Semantic markers* are domain-dependent word patterns and must be defined for a given corpus. They identify *cue-words* serving both as *separators* among logical subparts of the same sentence and as *introducers* of *semantic constituents*. In our specific case they allow us to search for the content of the query only in interesting parts of the sentence. One of the most important separators is the *announcement-query separator*. The LHIP clauses defining this separator can be one or more words covering rule like for instance:

```
ann_query_separator([X],0.7) #1.0 ~~>
        @terminal(X),
        {X='téléphone'}.
ann_query_separator([X,Y],1) #1.0 ~~>
        @terminal(X),
        @terminal(Y),
        {[X = 'numéro',Y = 'de']}.
```

As an example of semantic constituents introducers we propose here the following rule:

```
street_intro([T,Prep],1) #1.0 ~~>
        * street_type(T),
        preposition(Prep).
```

which make use of some word knowledge about street types coming from an external thesaurus like:

```
street_type(X) ~~>
        @terminal(X),
        {thesaurus(street,W),member(X,W)}.
```

It should be noted that we mix weighted and non-weighted rules, simply because non-weighted rules are rules with the highest weight 1.

2.2.1 Generation of hypotheses

The generation of annotation hypotheses is performed by: composing weighted rules, assembling constituents and filtering possible hypotheses. In this case the grammar should provide a means to provide an empty constituent when all possible hypothesis rules have failed. The highest level constituent is represented by the whole sentence structure which simply specifies the possible orders of constituents relative to annotation hypotheses.

```
s(s([Ann,Query]), W) ~~>
                 ann(Ann),
                 query(Query,W2).

ann(ann(Ann)) ~~> closK(word(Ann)).

query(query(Q),W) ~~>
          * ann_query_separator(QSep,W1),
           target(Target,W2),
           address(Addr,W3)
           {app_list([Qsep,[Target],
           [Addr]],Q),
            min_list([W1,W2,W3],W)}.
```

In the *ann* rule we have made use of the Kleene closure operator *closK* which allow LHIP to simply formulate regular expressions. In the *query* rule we have specified a possible order of constituents interleaved by semantic markers (e.g. separators and introducers). In this case we did not provide any linguistic constraint (e.g. preferring names belonging to the minimal common syntactic sub-tree or those having the longest sequence of proper names belonging to the same sub-tree).

3 Conclusions and future works

In this paper we summarized a proposal for a framework for designing grammar-based automated annotation applications. Starting with a case study and following an approach which combines the notions of fuzziness and robustness in sentence parsing, we showed how to build practical domain-dependent rules which can be applied whenever it is possible to superimpose a sentence-level semantic structure to a text without relying on a previous deep syntactical analysis. Even if the query generation problem may not seem a critical application one should bear in mind that the sentence processing must be done on-line.

As we have previously seen, the cue-words used as semantic markers are domain-dependent. Even their relevance disposal and their weight within the rules depends on their linguistic usage. Therefore, a complete automatic annotation system based on the approach proposed in this article seems to be adequate to give precise results. However, a semi-automatic system could satisfy our needs. This system should be based on the following techniques to achieve a high level of performance:

1. For each annotation, the system offers a list of propositions based on standard grammars as well as on external knowledge (ontologies, knowledge bases ...)
2. According to the grammar initially proposed, the user may change the annotation according to his needs. These modifications are held within the system to change the grammar rules as well as their weights. This makes the system interactive and enhanced by a learning phase.
3. We could imagine that rule design process can be partially automated and we intend to pursue some research on developing methods for both assisted rule design and corpus based rule induction.

References

Peter. R.J. Asveld. 1996. Towards robustness in parsing - fuzzifying context-free language recognition. In J. Dassow, G. Rozemberg, and A. Salomaa, editors, *Developments in Language Theory II - At the Crossroad of Mathematics*, Computer Science and Biology, pages 443–453. World Scientific, Singapore.

A. Ballim and G. Russell. 1994. LHIP: Extended DCGs for Configurable Robust Parsing. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 501 – 507, Kyoto, Japan. ACL.

R. Basili and Pazienza M.T. 1997. Lexical acquisitiion and information extraction. In Pazienza M.T., editor, *Information Extraction – A multidiciplinary approach to an ermerging information technology*, volume 1299 of *LNAI*, pages 44–72. Springer Verlag.

J-C. Chappelier, M. Rajman, P. Bouillon, S. Armstrong, V. Pallotta, and A Ballim. 1999. Isis project: final report. Technical report, Computer Science Department - Swiss Federal Institute of Technology, September.

G. Grefenstette. 1996. Light parsing as finite-state filtering. In Kornai A., editor, *Proceedings of the ECAI 96 Workshop on Extended Finite State Models of Language*, pages 20–25.

J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. 1996. Fastus: a cascaded finite-state transducer for extracting information from natural-language text. In E. Roche and Y. Schabes, editors, *Finite State Devices for Natural Language Processing.* MIT Press, Cambridge MA.

E.T. Lee and L.A. Zadeh. 1969. Note on fuzzy languages. *Information Science*, 1:421–434.

C. Lieske and A. Ballim. 1998. Rethinking natural language processing with prolog. In *Proceedings of Practical Applications of Prolog and Practical Applications of Constraint Technology (PAPPACTS98)*, London,UK. Practical Application Company.

D. Moll, J. Berri, and M. Hess. 1998. A real world implementation of answer extraction. In *Proc. of the 9th International Conference and Workshop on Database and Expert Systems. Workshop on Natural Language and Information Systems*, volume NLIS'98, pages 143–148, Vienna.

C.M. Sperberg-McQueen and L. Burnard, editors. 1994. *Guidelines for Electronic Text Encoding and Interchange, Text Encoding Initiative.* Chicago and Oxford.