A Two-Level Strategy for Parsing Procedural Texts.

Estelle Delpech, Murguia Elizabeth, Patrick Saint Dizier

A Two-Level Strategy for Parsing Procedural Texts

Estelle Delpech, Elixabete Murguia, Patrick Saint-Dizier
IRIT - CNRS,
118, route de Narbonne,
31062 Toulouse Cedex 9, France
delpech_estelle@yahoo.fr, emurguia@yahoo.com, stdizier@irit.fr

Abstract

This paper presents ongoing work dedicated to parsing the textual structure of procedural texts. This work is based on a two level approach: first, basic structures in texts are segmented using automata techniques and then a text grammar is applied to assemble these fragments. The grammar is based on X-bar syntax, transposed to text structure.

1 Credits

We thank the ANR-RNTL program trough the TextCoop project for supporting this research.

2 Introduction

The main goal of this work is to be able to answer procedural questions, which are questions whose induced response is typically a fragment, more or less large, of a procedure, i.e., a set of coherent instructions designed to reach a goal. Recent informal observations from queries to Web search engines tend to show that procedural questions is the second largest set of queries after factoid questions (de Rijke, 2005). This is confirmed by another detailed study carried out by (Yin, 2004). Procedural question-answering systems are of much interest both to the large-public via the Web, and to more technical staff, for example to query large textual databases dedicated to various types of procedures, including economic and commercial procedures (legal, know-how, etc.).

Answering procedural questions thus requires to be able to extract not simply a word in a text fragment, as for factoid questions, but a well-formed text structure which may be quite large. Thus, the techniques used for factoid questions do not seem adequate to deal with the problem at hand. We propose that a different approach should be adopted. We propose that the use of text grammars is a more appropriate and precise manner for representing and recognizing procedural knowledge in a text.

Analysing a procedural text requires a dedicated discourse grammar. Such grammars are not very common yet due to the complex intertwinning of lexical, syntactic, semantic and pragmatic factors they require (see e.g. functional discourse grammars and systemic grammars) to get a correct analysis. Discourse grammars have basically a top-down organization, they take discourse acts as their basic units, instead of just words, they account for the structure and the interactions between these acts and they require a relatively elaborated conceptual representation as output. Such a grammar must capture the discourse cohesion, including the communicative intentions as well as the discourse organization, e.g. in terms of plans.

Procedural texts explain how to execute procedures. In our perspective, procedural texts range from apparently simple cooking recipes to large maintenance manuals (whose paper versions are measured in tons e.g. for aircraft maintenance). They also include documents as diverse as teaching texts, economic regulations and know-how texts, medical notices, social behavior recommendations, directions for use, advice texts, savoir-faire guides etc. Even if procedural texts adhere more or less to a number of structural criteria, which may depend on the author's writing abilities and on traditions associated with a given domain, we observed a very large variety of realisations, which makes parsing such texts quite challenging.

Procedural texts explain how to realize a certain goal by means of actions which are at least partially temporally organized. Procedural texts can indeed be a simple, ordered list of instructions to reach a goal, but they can also be less linear, outlining different ways to realize something, with arguments, advices, conditions, hypothesis, preferences. They also often contain a number of recommendations, warnings, and comments of various sorts. The organization of a procedural text is in general made visible by means of linguistic and typographic marks. Another feature is that procedural texts tend to minimize the distance between language and action. Plans to realize a goal are made as immediate and explicit as necessary, the objective being to reduce the inferences that the user will have to make before acting. Texts are thus oriented towards action, they therefore combine instructions with icons, images, graphics, summaries,

preventions, advices, etc.

Research on procedural texts was initiated by works in psychology, cognitive ergonomics, and didactics. Several facets, such as temporal and argumentative structures have then been subject to general purpose investigations in linguistics, but they need to be customized to this type of text. There is however very little work done in Computational Linguistics circles. The present work is based on a preliminary experiment we carried out (Aouladomar and Saint-Dizier 2005), where a preliminary structure was proposed.

From a methodological point of view, our approach is based on (1) a conceptual and linguistic analysis of the notion of procedure and (2) a mainly manual corpus-based analysis, whose aim is to validate and enrich the former. The originality of our work lies at the level of the objects parsed in discourse, and on the way linguistic principles and norms are inserted to deal with the large variety of forms procedural texts may have. The paper is structured as follows: we first present our generative approach to text processing, we then develop the two steps of the treatment: segmentation and grammar application.

3 Text Grammars

The answer to a procedural question is a well-formed fragment of a text, it might include a sequence of *n* instructions linked by various markers (e.g, coordinators, temporal marks) or typographical cues (e.g., comma, dot, newline). Thus, the answer cannot be found locally, like in the case of factoid questions.

3.1 The Structure of Procedural Texts

The structure of procedural texts can be more or less complex. There are certain types of procedural texts (e.g. recipes) which are not very structured, in other words, what we generally find is a goal (e.g. *How to cook a paella*) followed by a sequence of steps to follow in the form of instructions for the user. Nevertheless, this is not always the case. In some other texts (e.g. do-it yourself, maintenance manuals), the structure is not so flat, what we find is a more hierarchized division of the text in subprocedures to reach the goal. For example, a text about *Maintainance of your swimming pool* can be divided in the following subsections: *Emptying the swimming pool, Cleaning the swimming pool, Mantainance after cleaning and control of the pH level, Avoid the formation of fungus*. All of these subgoals are followed as well by a sequence of instructions. In general, we can say that the longer or more complex the procedure is, the more divisions the text is going to have.

In (Aouladomar and Saint-Dizier 2005) an analysis of procedural texts is presented. This work identifies different elements, among them: instructions, goals, warnings, prerequisites, and pictures. We assume this classification for this work. Among these building elements, the central unit is Instruction, together with Goal. They both constitute the core of a procedural text. This work showed that a constituent in a procedural text may occupy a large variety of positions, and a number of them may be optional. Writing a text grammar of this kind results in a set of rules with little constraints and with a low predictive and explanatory power. To circumvent these problems to a certain extend, we explored the possibility of defining a set of principles, constraints and norms that would account for procedural text structure in a better way. We borrowed some of these principles to Generative Grammar. This led to the development of specific parsing strategies, presented hereafter.

3.2 A Grammar for Procedural Texts

In order to define our grammar, we borrow some notions from generative grammars and transpose them to the problem we have in hand. For example, we adopt the notion of head which is the central element in a syntactic phrase, and combines with other elements to the right (a complement) and to the left (specifier) to project structure. We transfer X-bar syntax to text structure. We also borrow the distinction between complements and adjuncts, or obligatory and optional material. Depending on whether an element belongs to one category or the other, it occupies a different position in the structure. We also aim at a slight shift from 'classical' grammar descriptions to a principled-based approach, using principles, parameters, but also norms that capture the different types of realizations observed over genres and domains, used as heuristics (e.g. average length of titles and instructions, most frequent verb classes used, etc.). A clear di-

vision between central and peripherial (or obligatory and optional) elements is made and reflected in the final structure. Also, the relations between the different building blocks in a text (e.g. scope relations) are more clearly delimited, since the resulting structure is not flat but rich in terms of hierarchical structure.

We conceive the structure of a procedural text as the projection of a Goal (i.e. the title of the text) and its complement which can either be a sequence of instructions or a sequence of tasks. Thus, the root grammar rule is:

```
Procedure --> Goal seqInstr
Procedure --> Goal seqTasks
```

We continue defining our grammar from the lower level units up to the root grammar rule. For the purpose of this work, we assume that the basic unit is the Instruction. Instructions, however, do not come in isolation, but as we already said combined with other instructions. We call these combinations sequences of instructions (seqInstr), and we analyze them as a coordination of instructions, where the head of the phrase is a sequence marker. We include below the rules that identify this type of structure:

```
seqInstr --> Instr seq'
seq' --> seqmarker Instr
seq' --> seqmarker seqInstr
```

If the first and the second rule apply, a structure like the following could be recognized: *Hold the bracket down on the tabletop, then screw one of the screws through the hole in the bracket*. The third rule takes care of recursivity, and it will recognize any sequence formed of more than two instructions, such as: *Turn on the access point, and activate the wireless connections for your devices. Verify that they are all transmitting a wireless signal....*

A sequence of instructions is the complement of a Goal. Thus, it is obligatory (we cannot have a Goal without a seqInst associated, it will result in an ill-formed structure). Goals are the heads of their phrase, they project up a Task. In the specifier position of this phrase, we can optionally find some prerequisites (e.g. the ingredients in a recipe).

```
Task --> (spec) G'
G' --> Goal seqInstr
G' --> Goal seqTasks
```

With the first and second rule, we can recognize structures such as (i) *Fixing the top rails to the table top*. Place the table top upside down on an even floor, saw stools or workbench. Position the top rails on the underside of the table top ... and (ii) *To get dried oil paint off a brush, soak it in nail-polish remover for a minute or two, after that you wipe off what remains and clean your brushes with soap and water as usual*. In the first example the Goal is the first sentence, which is a subtitle. This is followed by its complement, a sequence of instructions, and together they project a Task. In the second example, the Goal is the preposed purpose clause introduced by *To*, which is also followed by a sequence of instructions. The third rule describes the situation where the complement of a Goal is a sequence of tasks rather than of instructions. An example of this type of structure is the one provided in the previous section of *Maintainance of a swimming pool*, where the main procedure is divided in different subprocedures or subtasks, each followed by a seqInstr, building a seqTasks. We analyze sequences of Tasks in the same way as sequences of instructions. The first and second rule represent basic cases, the third recursivity:

```
seqTasks --> Task seq'
seq' --> seq Task
seq' --> seq seqTasks
```

Optional elements are adjoined to different levels of the structure, depending on their scope. For example, we can have a condition linked to one instruction as in *If it is not soft, boil it for ten more minutes*, but also adjoined to a sequence of instructions *If you have Windows 95 installed in your computer, turn on the computer and follow the instructions on the screen*. Other elements like warnings or pictures can also be adjoined to those nodes, or to Tasks or sequences of tasks. The rules for adjuncts follow, adjuncts can either appear to the right or the left, that is what the symbol + codes:

```
Instr --> Adjunct + Instr
seqInstr --> Adjunct + seqInstr
Task --> Adjunct + Task
```

```
seqTasks --> Adjunct + seqTasks
```

4 Parsing with Text Grammars

The grammar defined above identifies text structures relevant for answering procedural questions. Obviously, we do not aim at making a full parse of those texts. We need to identify 'terminal' structures (such as titles and instructions), and then to assemble them together.

The parsing of procedural texts is realized in two steps. First a segmentor, comparable to a lexical tagging component for sentence words, identifies basic structures and then the X-bar grammar presented above assembles these basic structures. The segmentor makes an initial proposal to the grammar. In case of a failure of the grammar, backtracking is possible, but to a limited extend, so that relatively optimal segmentations are kept.

The starting point of our system are web pages, these are cleaned so that only relevant tags for the segmentor or the grammar are kept. They are then tagged with the TreeTagger, in order to incorporate category and relevant morphological information. A final step identifies the semantic class of the verbs, according to a classification we carried out, largely inspired from WordNet classification. Our classification is composed of a hierarchy of 186 classes, among which: verbs of change, verbs of cleaning, maintenance, etc.

4.1 The segmentor

The segmentor has several goals:

- First, to tag terminal discourse elements: instructions, titles (viewed as the expression of goals), warnings stated outside instructions, prerequisites, some forms of arguments and connectors. This is useful to reduce the non-determinism in the grammar.
- Via the tagging, to allow for the identification in a large text of zones which are more procedural than others (large texts may be verbose and contain non procedural elements such as comments or historical considerations), allowing then to focus the search of responses on a certain text area.

It is clear that the form of those terminal objects and the criteria required to define them largely varies over application domains, textual genres and the targeted audiences. Our strategy was to define several sets of criteria, valid for a group of domains that share common discourse forms for describing procedures. Our approach was to proceed by 'domain aggregation'. For example we first considered samples from coocking recipes texts, and defined the segmentation criteria. Then, we considered other domains which turn out to have a close structure: 'do it yourself' and video game solutions. At a certain stage, we get a stable set of criteria which can be implemented as an automaton. We plan to have in the end a small number of automata, each encoding the discourse structure of a group of domains.

Finally, each type of object we have to segment requires a different approach of segmentation, because the identification criteria are very different. We briefly present them below.

Dealing with Instructions

As far as instructions are concerned, the model is relatively straightforward finite state automaton, with some little tricks, since some marks may be common to two adjacent instructions, such as punctuation. The main marks for instructions are:

1. typographic and punctuation marks, and temporal marks, used for delimiting instructions,
2. verbs, together with their morphology (basically infinitive and imperative forms are quite frequent incoocking recipes, but other tenses are found in other domains), and their semantic class (action verbs and subclasses) since semantic class may be used to identify different types of instructions; deverbals and predicative nouns are also relevant,
3. various classes of marks, such as: modal marks ('you must do'), reminders ('do not forget to'), performance marks ('care about doing'), marks describing optionality or advices ('it is preferable to'), injunctive forms, and adverbs of manner.

Besides these elements used to identify instructions, instructions may contain a lot of elements such as: low level goals, pictures, local warnings, hyperlinks, etc.

Recognizing Titles

Recognizing titles is much more challenging. They have in general the form of an instruction (e.g. mounting your computer), but with a different layout. Recognizing titles is crucial for answering questions.

Titles are first identified by the typography: bold font, possibly underlined, or via the use of dedicated html marks (h1, etc.). Finally, we can also rely on the level of generality of the verbs used in titles, which are more generic than those found in instructions (we use the Volem verb base for that purpose). In fact, titles can be viewed as 'super-instructions', this distinction being highly domain dependent.

Prerequisites as well as warnings may also have titles. However, these two latter objects have a different typology (although they may also contain instructions), which allows us to make the distinction among types of titles, and to isolate those effectively governing instructions, to be interpreted as denoting goals.

A second problem is to identify the hierarchy of titles, which is important in most texts of a certain length. Identifying such a hierarchy allows us to associate more precisely sequences of instructions to a goal. Since dedicated html tags are not so frequent to discriminate titles, we must rely on other factors, among which:

- presence of capital letters, or size in number of words (quite frequently 4 to 5 words, with no pronominal references),
- level of the verbs in titles: higher titles contain more generic verbs,
- identification of islands of instructions which share a quite large number of common words (entailing a certain thematic cohesion of instructions below a title, as in Centering Theory).
- identification of summaries or introductions below titles which contain words present in subtitles.

This is however still a very tentative option.

Another kind of difficulty is that titles are often elliptic (e.g. the verb is missing). In some situation they may be just absent, therefore, we may need, to answer questions, to be able to reconstruct these.

Dealing with Warnings

Warnings as well as arguments are introduced by a range of specific verbs often in the imperative form or by negative connectors, which is quite easy to identify in French. Here are a few examples:

- negative connectors: *sous peine de, sinon, car sinon, sans quoi,* etc. (otherwise, under the risk of),
- risk verbs: *risquer, causer, nuire, commettre,* etc.
- prevention verbs: *éviter, prévenir,* etc.
- negative expressions: *de facon à ne pas, pour ne pas, pour que ... ne ...pas,* etc. (in order not to).

Indentifying Discursive marks

Temporal marks are the most frequent marks, they include: precedence, overlap, inclusion, parallelism, etc. They are mainly realized by means of adverbs, prepositions, conjunctions, aspectual verbs and propositions describing the realization of an event. Marks are annotated by the TreeTagger and typed via a predefined list we have elaborated.

Causal marks are particularly rich and diverse. They are used to relate a goal to a set of instructions, or to specify within an instruction its aim; causal marks are also used to identify objectives, warnings and various forms of preventions, consequences and some forms of conclusions.

Besides these two main classes of marks, we noted a few conditionals and alternative marks. These are often prepositions or semantically closely related to the semantic typology specific of prepositions. To identify and interpret them, we use the PrepNet framework (www.irit.fr/recherches/ILPL/prepnet.html).

Global Architecture

The automaton first recognises instructions, then titles, warnings and prerequisites. Segmentation is confronted to several difficulties, among which:

- ambiguities of various sorts: coordination or punctuation in instructions: for that purpose we must develop some very local grammars that check the status of these elements and decide

whether they are inside an instruction or must be considered as a delimitation mark.

- several partially overlapping structures candidates to be an instruction, this is the case for instructions that contain e.g. warnings or advices that may or may not be also considered as separate instructions. At this stage, we can recognize them, but keep them in a single instruction.
- shared items between adjacent instructions, entails some forms of either partial parsing (making e.g. the hypothesis that an instruction may lack a begining mark) or stacks to keep track of elements which may be shared (e.g. shared arguments in ellipsis).
- a few unexpected forms, e.g. using the future tense.

The result of the segmentor is a representation of the following form (simplified for readability):

```
<procedure>
<title> poser une tringle  rideaux </title>
<warning> attention a ne pas perdre des elements
       </warning>
<prereq> les regles de base .... </prereq>
<seqinstr>
<warning> disposez de suffisemment d'espace
       </warning>
<instr> 1. tracer la hauteur de la tringle,
         </instr>
<instr> 2. couper la tringle a la bonne
                   longueur. </instr>
etc.
</seqinstr> </procedure>
```

4.2 The grammar implementation

The grammar is used to bind the elements identified by the segmentor. It gives a global structure to the text, which is necessary and sufficient to answer *how-to* questions. One of its goals is to give a precise scope to each element. It also binds adjuncts at appropriate places, giving these a priori a minimal attachment.

The grammar produces an XML output, usable for the search of responses. In particular, titles, viewed as goals, are considered for unification with the question body. Some instruction fragments may also be considered to identify subtopics. XML tags therefore identify indexes used by the question-answering system.

We have designed a prototype in Prolog, implementing a top-down engine. In spite of its relative inefficiency, we have adopted so far an interpreter mode so that every step of the system can be evaluated. Another goal of this first implementation is to be able to add heuristics quite easily and to test their impact. These heuristics capture the notion of norm introduced above; they contribute to resolving ambiguities and to compensate for structures not recognized by the segmentor. For example, we have:

- resolving ambiguous attachment of constructions: a typical case is the attachment of pre-requisites, which can be bound to a number of specifier positions: our norm is to have a maximal attachment, i.e. to have prerequisites attached as high as possible in the tree. An opposite case are warnings, which need to be as close as possible to instructions: they undergo a minimal attachment, being adjoined at the lowest level possible.
- going over structures not recognized by the segmentor: a typical situation is when the segmentor does not have recognized a title (it is incomplete, illformed w.r.t. the segmentor, or it does not exist and needs to be infered for answering questions). Each domain being associated with a small number of norms (average instruction size, average number of instructions per task, per sequence of instructions, etc.), when for example, an unexpectedly large number of instructions is found without any structure binding it (sequence of instruction tag or title), our norms allow us to suggest that, based on specific typographical marks, a title has possibly been skipped at a certain position.

Finally, the engine is flexible enough so that partial parses can be carried out, especially on large texts, where there is a lot of useless text or where there are some risks of failure.

5 Perspectives

This short paper relates ongoing work on parsing procedural texts via the pairing of a segmentor, capable of recognizing, basically, terminal text structures, and a grammar, inspired from generative principles, that better captures the variability of procedural styles over domains and authors.

The implementations proposed so far are preliminary and allow us to explore the various types of problems one may encounter when dealing with text grammars. The corpus considered is of a rather modest size, but with quite diverse structures. It is a development corpus, allowing us to better analyse the behavior of the different components we have developed. Outputs are checked manually: this is a quite challenging task, since most texts include more than one hundred tags.

Since this project is designed to become an industrial project, it is clear that we need to investigate other forms of implementations once the linguistic quality of the segmentor and of the grammar is satisfactory. A possible direction is to develop learning mechanisms trained on the outputs of our parser. These would produce probably slightly less good results, but would allow for the processing of large amounts of texts over a limited period of time.

References

[1] Aouladomar, F., Saint-Dizier, P., *An Exploration of the Diversity of Natural Argumentation in Instructional Texts*, 5th International Workshop on Computational Models of Natural Argument, IJCAI, Edinburgh, 2005.

[2] Delin, J., Hartley, A., Paris, C., Scott, D., Vander Linden, K., *Expressing Procedural Relationships in Multilingual Instructions*, Proceedings of the Seventh International Workshop on Natural Language Generation, pp. 61-70, Maine, USA, 1994.

[3] Kosseim, L., Lapalme, G., *Choosing Rhetorical Structures to Plan Instructional Texts*, Computational Intelligence, Blackwell, Boston, 2000.

[4] De Rijke, M., *Question Answering: What's Next?*, the Sixth International Workshop on Computational Semantics, Tilburg, 2005.

[5] Hovy, E., Hermjakob, D., Ravichandran, D., *A Question/Answer Typology with Surface Text Patterns*, Proceedings of the DARPA Human Language Technology Conference (HLT), San Diego, 2002a.

[6] Maybury, M., *New Directions in Question Answering*, The MIT Press, Menlo Park, 2004.

[7] Moldovan, D., Harabagiu, S., Pasca, M., Milhacea, R., Goodrum, R., Grju, R., Rus, V., *The Structure and Performance of an Open-Domain Question Answering System*, Proceedings of the 38th Meeting of the Association for Computational Linguistics (ACL), Hong Kong, 2000.

[8] Yin, L., *Topic Analysis and Answering Procedural Questions*, Information Technology Research Institute Technical Report Series, ITRI-04-14, University of Brighton, UK, 2004.