An algorithm for generating document-deictic references

Ivandré PARABONI
ITRI, University of Brighton
Lewes Road
Brighton, UK, BN2 4GJ
ivandre.paraboni@itri.brighton.ac.uk

Abstract

This paper will focus on how a natural language generation system can produce references to parts of documents such as sections, pictures paragraphs *etc.*, a phenomenon we call *document deixis*. An example is the definite description "picture 3 in section B". We propose an algorithm to be implemented as an extension of an existing NLG system, which is able to generate structurally complex documents, allowing the production of document deictic references for different purposes, such as helping the reader to find a particular piece of information or simplifying the reference to domain entities.

Introduction

Many documents are organised in hierarchically-structured components (e.g., pictures, sections, subsections, itemised lists *etc.*), and such components may be *referred to* within the document (e.g., "read section 1.2.") for various purposes, such as to emphasise relevant parts of the document, to link text and corresponding pictures, to simplify the reference to domain entities described somewhere else in the document, *etc.* Cross-media references of this kind are a prime example of the phenomena influencing the coherence of textual or multimedia documents, e.g., André and Rist (1994), McKeown et al (1992).

We call the referable parts of a document *document entities* (DocEnt), and we call a reference to a DocEnt *document deixis*[1] (ddx). We found substantial amounts of ddx in technical manuals, legislation, scientific articles *etc*. Our own research focuses primarily on references to DocEnts found in medical patient information leaflets or PILs corpus in ABPI (1997) and, more specifically, on how documents containing such descriptions can be generated from a medical knowledge base.

In this paper we will discuss how the problem of generating ddx references differs from the generation of other forms of definite descriptions (e.g., references to domain entities). The problem of deciding *when* to produce such references has been described elsewhere - see Paraboni and van Deemter (1999). The deictic aspect of such references (where the position of the referring expression itself has to be taken into account) and the hierarchical structure of the domain make the generation of ddx references an ideal area for investigation of some little explored aspects of reference.

1 Basic concepts

1.1 Document structure and layout

We will consider as an example a hierarchical document structure made up of terminal DocEnts of type *text* and *picture* representing the textual and

[1] The use of the term "document deixis" parallels the use of "discourse deixis" in Webber (1991): we say "deixis" because the expression relates utterances to the spatio-temporal co-ordinates of the act of utterance cf. Lyons (1977), and we say "document" because those entities are defined in the document structure (and not in discourse, as the phenomena addressed by Webber).

graphical components of the document, and non-terminal DocEnts *doc*, *section*, *part*, *list* and *item* representing its hierarchical levels. A simple example of a document constructed according to this structure is shown in Figure 1. Although not discussed in this paper, the present specification could include further referable document parts such as paragraphs, sentences, words *etc*., which would be in principle equally supported by the generic algorithm presented in section 2.

A reference to a DocEnt `r` can make use of a relevant *layout attribute* to which a particular *value* is assigned so as to distinguish the referent from other nodes of the structure. Layout attributes include 'section_number', 'item_label' etc. For example, a given section of the document may be described by its section number (the attribute of the reference), to which the value "1" is assigned as in "section 1". Following Dale and Reiter (1995) we call such pair (attribute, value) as in (section_number, "1") a *property*. For simplicity, we will concentrate on references which make use of layout attributes such as numbers, titles *etc*., although the principles under discussion are applicable to many other forms of reference, such as the use of spatial (e.g., "the previous section") and ordinal (e.g., "the 1st section") attributes as discussed in Paraboni (2000).

Layout attributes of a given DocEnt type `t` are assigned values which are meant to uniquely distinguish those DocEnts of type `t` *within a certain subtree of the document structure.* For example, item labels (a layout attribute of DocEnts of type *item*) are assigned the distinguishing values ("A", "B", "C") within the subtree rooted in the relevant (list) node. In our example, we assume that the relevant layout attributes of the DocEnt types in Figure 1 are assigned distinguishing values as follows:

- DocEnts of type *section* are assigned distinguishing values (section_number = 1...2) throughout the document, i.e., within the (sub)tree rooted in (doc);
- DocEnts of type *part* are assigned distinguishing values (part_label = A...B) per section, i.e., within the subtree rooted in each relevant (section) node;
- DocEnts of type *item* are assigned distinguishing values (item_label = A...C) per itemised list, i.e., within the subtree rooted in each relevant (list) node and
- the enumeration of DocEnts of type *picture* (either throughout the entire document or per section) will be specified as required by the various examples discussed in this paper.

We call the *scope* of a given layout attribute `a` associated with DocEnts of a given type `t` the subtree within which `a` assigns distinguishing values for each node of type `t`. For example, the scope of the layout attribute 'item_label' associated with DocEnts of type *item* is the subtree rooted in the corresponding (list) node. Scopes of layout attributes are inferred from the existing document layout and will be a key element of the algorithm presented in section 2.

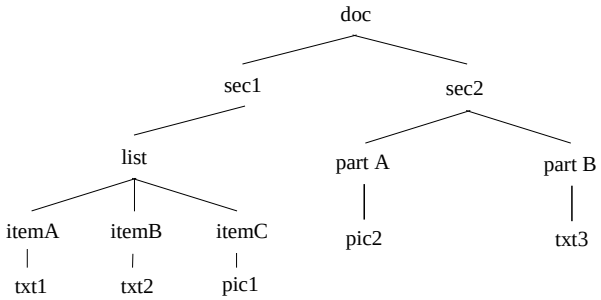

Figure 1 - Example of document structure

1.2 Document-deictic descriptions

In a document structure as exemplified in Figure 1, the *text* nodes represent the textual components of the document. Since we are considering a document that refers to *its own* parts, some of the existing *text* nodes may in fact contain ddx descriptions. For a given reference to any node r of the document, we call d the *text* node which contains the corresponding referring expression. For example the *text* node (d = txt2) may contain the description "item C", which is a reference to the node (r = itemC) of the document.

In its simplest form, a given reference (d, r) can be represented as a description {r, a, v}, where a is a layout attribute associated with all the DocEnts of the same type as r, and v is the current value of a taken from the layout of the document. The pair (a, v) represents a property of r.

The surface realisation of a description may denote more than one node in a given document structure. For example, in a document structure containing several lists of enumerated items, the description "item A" may denote several *item* nodes. Analogous to Dale and Reiter (1995), we call *distractors* of a description {r, a, v} the set of all DocEnts of the same type of r in the structure which are not r itself, and whose attribute a is assigned the same value v.

The use of a particular property (a, v) in a description may distinguish the referent r from some or all its distractors. Accordingly, algorithms such as presented in Dale and Reiter (1995) for generating referring expressions make use of properties in order to rule out distractors and produce uniquely distinguishing descriptions.

However, apart from providing uniquely distinguishing descriptions of referents, ddx descriptions also have a second purpose: ddxs serve to *locate* the referent in the document, and for that reason the uniqueness of a description does not suffice as a condition for its felicity. For example, assuming (d = txt3, r = itemA) in the document presented in Figure 1, the description "item A", even though being a uniquely distinguishing description of r, is misleading in two senses: (i) the reader of the referring expression in d cannot be assumed to know *where* in the hierarchical structure of the document the referred item is to be found and (ii) the reader cannot be assumed to know that there is only one "item A" in the whole document. We say that a ddx reference which is misleading in one of the senses (i) or (ii) is incomplete. For the reference to be complete, the layout and structure of the document have to make it clear what is the part of the document within which the property is to be interpreted (i.e., where the actual distractors of the reference are).

One way in which appropriate ddx descriptions can be generated is by taking into account, instead of the set of distractors ruled out by the description {r, a, v}, only the set of distractors *within the scope* of a. More specifically, a description {r, a, v} is said to be *complete* if the scope of a is a subtree which contains the node d itself. For example, "item A" is a complete description of (d = txt2, r = itemA), since the scope of the layout feature (the item_label attribute) is a subtree rooted in the (list) node, which contains the node d as discussed in section 1.1. On the other hand, "item A" is not a complete description if we consider (d = txt3, r = itemA), since the scope subtree in this case does not contain the node d.

As shown by the examples, the completeness of a ddx reference depends on the position of the referring expression d in the document. This will be reflected by the algorithm presented in the next section for the generation of ddx references which are both distinguishing and complete.

2 Algorithm description

Given the present definition of completeness, the task of the algorithm is to decide how to proceed in case a given reference is not complete. This can be done by adding further levels of description (namely, references to ancestor nodes of the referent) as in "item A of the list in section 1". More specifically, a ddx (d, r) will be realised as a series of ($n > 0$) hierarchically ordered pairs in the form $\{(x_1, a_1, v_1)...(x_n, a_n, v_n)\}$, where x_1 is r itself, every x_j for ($j > 1$) is a selected ancestor node of x_{j-1} and every pair (a_j, v_j) is a layout property (i.e., a pair attribute, value) of the corresponding x_j.

The number n of descriptions required to produce a complete description in this way depends on the scope subtree of each attribute a_j. The algorithm initially attempts to produce a description of (x_j = r) for (j = 1). For each description, an appropriate pair (a_j, v_j) is selected according to some defined criteria[2] and once the scope of a_j is a subtree which contains the node d itself, a complete description has been obtained and the process finalises. Otherwise, a reference to a selected ancestor node of x_j will be included in the resulting description. More specifically, x_{j+1} will be the root node of the scope subtree of the *previous* reference attribute, i.e., a_j. Although in most cases x_{j+1} will coincide with the immediate parent node of x_j, the node to be selected has to be defined as the root of the scope of a_j in order to avoid intermediate redundant levels of description. For example, assuming that pictures are enumerated from 1 at the beginning of each individual section in Figure 1, for (d = txt1, r = pic2), after producing the (incomplete) description of (x_1 = pic2), the next level to be described is (x_2 = sec2), since the scope of the picture number is a subtree rooted in (sec2), and not (part A), whose description in this case would be redundant.

Individual references are added to the resulting description in hierarchical order, from lowest to highest level, making the scope of the reference increase at each level, up to the point where the ddx description turns out to be complete and, consequently, unique, since the scope of a complete ddx description is a subtree containing the node d within which r can be uniquely identified.

The complete example of reference (d = txt1, r = pic2) can be illustrated as follows: assume pictures are enumerated separately in each section, that is, both (pic1) and (pic2) are labelled "picture 1" in the document of Figure 1. The algorithm starts by referring to (x_1 = pic2) and since in this case the scope of picture numbers is a subtree rooted in (sec2), it does not include d, hence "picture 1" is not (yet) a complete ddx. The next step is to produce a reference to the root of the scope subtree of this reference, which has to be, in this case, (x_2 = sec2). Since the scope of section numbers is a (sub)tree rooted in (doc), it includes the referring expression d, so the description becomes complete as "picture 2 in section 2". If pictures were enumerated from "1" throughout the document, the resulting complete ddx description would be simply "picture 1", as the scope of such picture number is a (sub)tree rooted in (doc), which contains the node d.

Since the algorithm stops when a complete reference is obtained, the resulting description does not include references to any redundant upper level. For example, the algorithm does not produce "part A of section 2" if "part A" alone suffices as a complete description of the referent. However, if desired in a particular domain, this form of redundancy can be easily allowed by taking into account, instead of the scope subtree s of a given attribute, a subtree of s. For example, in cases where pictures are enumerated throughout the entire document, the scope s of picture numbers is the tree rooted in (doc) and, consequently, references to such pictures will never include extra levels of descriptions. By pretending that pictures are enumerated per sections instead of throughout the entire document, i.e., considering a subtree rooted in the relevant (section) node instead of s, the algorithm will produce slightly redundant references such as "picture 1 in section 1".

In this paper we greatly simplified the algorithm, omitting, for example, its ability to produce references based on different types of attributes. However, it is worth noting that the choice for a particular (a_j, v_j) may lead to different lengths of descriptions. For example, in the document presented in Figure 1, the ddx (d = txt3, r = part A) could be realised as either one single reference (e.g., "part A", "the previous part" *etc*.) or two references (e.g., "the 1st part of this section"). The use of different types of properties and criteria for property selection aiming at *minimality* of descriptions are discussed in Paraboni (2000).

Conclusions and future work

In this paper we have discussed some aspects of the generation of document-deictic references, a phenomenon ubiquitously found in structurally complex documents made of textual and graphical components. We discussed how this task differs

[2] For a discussion on criteria for selecting properties for reference (e.g., "section 2" or "the previous section") see Paraboni (2000).

from the generation of ordinary definite descriptions and presented an algorithm for generating ddx descriptions which are both unique and complete. Although further constraints may be necessary in order to produce optimised references, the preliminary version of the algorithm is expected to generate most instances of the phenomena found in our corpus.

When completed, the algorithm will allow us to produce documents with artificially high numbers of ddx expressions. By comparing such documents with the corpus, we expect to precise how ddx references are actually used and, from the results of this investigation, we aim at expanding the current approach towards a theory of ddx generation.

Acknowledgements

This work is supported by the CNPq, the Brazilian Research Council.

References

ABPI - The Association of the British Pharmaceutical Industry (1997). 1996-1997 ABPI Compendium of Patient Information Leaflets.

Elisabeth André and Thomas Rist (1994). Referring to World Objects with text and Pictures. Proceedings of Coling'94.

Robert Dale and Ehud Reiter (1995). Computational interpretations of the Gricean maxims in the generation of referring expressions. Cognitive Science (19).

H.P. Grice (1975). Logic and Conversation. Cole and Morgan (Eds.) Syntax and Semantics, Vol.iii: Speech Acts. New York, Academic Press.

John Lyons (1977). *Semantics*. Cambridge University Press.

Kathleen McKeown et al. (1992). Generating Cross-References for Multimedia Explanation. Proceedings of the AAAI-92.

Ivandré Paraboni and Kees van Deemter (1999). Issues for the Generation of Document Deixis. In André et al (Eds.), Deixis, Demonstration and Deictic Belief in Multimedia Contexts. 11th ESSLLI, Utrecht, The Netherlands, pp.43-48.

Ivandré Paraboni (2000). Describing Document Parts. In: proceedings of the 3rd Cluk Computational Linguistics in the UK. Brighton, UK, pp. 34-41.

Bonnie Lynn Webber (1991). Structure and Ostension in the Interpretation of Discourse deixis. Language and Cognitive Processes 6(2) May 1991, pp. 107-135.