Document modelling for customised information delivery

Shijian LU, Cécile PARIS

CSIRO ICT Centre
Locked Bag 17,
North Ryde NSW 1670 Australia

(Shijian.lu, Cecile.paris)@csiro.au

Mingfang WU

CSIRO ICT Centre
Private Bag 33
Clayton South, VIC Australia

Mingfang.wu@csiro.au

Abstract *As the amount of information available to people multiplies at an increasing speed, it becomes ever more important to deliver information customised to users' specific needs. Natural Language Generation systems coupled with user modeling techniques have been built to address this issue, to produce information that is relevant to the users . A common approach adopted by such systems is an approach based on planning, starting from a discourse (or communicative) goal, and planning the text to be presented to the users. However, these systems are not easy to build and difficult to change by domain experts. One of the problems is that it is hard to specify the plans employed, because they often require knowledge about writing, domain expertise, knowledge of computational linguistics and, finally, knowledge about how to obtain data from the underlying information sources. In this paper, we present our first step to address this problem.*

Keywords Document modeling, information retrieval, document generation, personalized documents.

1.Introduction

The rapid advancement of information technology has made huge amount information available to more and more people. Increasingly, people depend on the availability of information to achieve their objectives. However, the availability of information does not necessarily translate to productivity gains. In fact, studies have shown that productivity often gets hampered as more and more people are suffering from information overload or fatigue [4]. Indeed, information must be relevant to one's information needs, and it must be easily understandable in order to be useful. That is where contextualised information delivery comes in. Studies have shown that documents tailored to the needs of individual users outperform general purpose documents, e.g., [8, 1, 11, 12].

However, applications delivering customised information are generally expensive to build. Broadly speaking, these applications fall into two categories: template-based and plan-based. There are pros and cons with either approach. Template-based systems are generally easier to construct, but harder to maintain and less flexible, while plan-based systems, with plans of finer level of granularity than typical templates are more flexible and can handle more situations, but require larger overhead to construct [9].

In this paper, we investigate why plan-based tailored document generation systems are difficult to build and report the result of our first step to mitigate the situation. In section 2, we provide some background information and our conception of the problem. Our approach is detailed in Section 3. Before concluding, an example of using this approach is provided in Section 4.

2.Background

Despite the fact that customised documents are often more effective than general purpose documents, applications for delivering tailored documents are far less common than they should be. A major reason is that such applications are not easy to develop. Taking plan-based systems as an example, there are at least two reasons why that is the case. First, these systems have typically required extensive semantic knowledge bases [10] which are expensive to craft. Second, the plan operators that underpin the systems' behaviour are difficult to construct.

In CSIRO, we have been developing Myriad [7], a platform for tailored information delivery. The Virtual Document Planner (VDP), its core component, exploits a plan-based approach based on More and Paris [6]. When we developed it, however, we paid particular attention to address the first issue above: we wanted the VDP to produce presentations customised for the user and the situation without the need for a large underlying semantic base. Instead, we wanted to exploit existing technology concerned with retrieving information from existing sources. As a result, the VDP combines discourse planning and document synthesis to gather information through the use of *retrieval services* (in this paper, the notion of *retrieval services* refer to software components which perform information retrieval functions.) that serve as the interface between the two. [2] This alleviates the need for an extensive (usually manually constructed) knowledge base.

The second issue remains: plan operators are typically difficult to write. Before we turn to this problem, we briefly discuss the advantages of using a plan-based approach, as opposed to a template-based one. With a

Proceedings of the 10th Australasian Document Computing Symposium, Sydney, Australia, December 12, 2005.

```
<operator>
   <id>tellUserAboutStaff<id>
   <effect>(Describe ?staff to ?user)</effect>
   <constraint>(user:isNewStaff ?user)</constraint>
          <nucleus>
                 <value>(inform ?user ?staff homepage)</value>
          </nucleus>
          <satellite>
                 <relation>context</relation>
                 <value>( inform?user ?staff team)</value>
          </satellite>
          <satellite>
                 <relation>context</relation>
                 <value>( inform?user ?staff project)</value>
          </satellite>
          <satellite>
                 <relation>elaboration</relation>
                 <value>( inform?user ?staff informationFromNet)</value>
          </satellite>
   </operator>
```

Figure 1. An example discourse plan operator

plan-based approach, a system starts with a communicative goal, and use discourse plan operators to decompose a high level goal into primitives, see More and Paris [6] for a more detailed description of the process. While doing so, the system builds *a discourse tree*, which is a rich source of information allowing the system to perform a number of reasoning tasks over the generated text.

To illustrate this particular point, let's take the recipe analogy. A recipe typically provides the sequence of steps to be done to produce the dish (i.e., the high-level goal). If something goes wrong (or if the specified ingredient is not available), the person following the recipe cannot reason about what went wrong (or about what other ingredient to use instead of the specified one), as s/he does not know what the purpose of the ingredient is and its role in the overall recipe. The only recourse is to find another recipe for the same dish, hoping this one will succeed (or to find a recipe that does not include that ingredient). Yet if the person understood the role of the ingredients and the steps, s/he may be able to understand what went wrong (or how to substitute another ingredient). Similarly, when producing a multi-sentential document, representing explicitly the intermediary goals and the relationships between the various chunks of information allows the system to understand the role of each element of information and to reason about their role in achieving the main communicative goal. This has been used, for example, to enable a system to participate in a dialogue [6] and to reason about to realise the text on the selected delivery medium [2]. It is because of the resulting discourse tree and the reasoning it enables that we chose the plan-based approach for our platform.

Discourse operators are the plans that tell the VDP how to plan the discourse, and, through their use and expansion, the discourse tree is generated. The plans in the VDP/Myriad include discourse goals and rhetorical relations, the latter based on Rhetorical Structure Theory (RST) [5]. Figure 1 is an example of a discourse operator. It specifies how the discourse goal is to be decomposed into subgoals, thus specifying what content is to be included in the text (at various levels of abstraction – e.g., "describe ?staff to ?user" and inform ?user of specific sub-topics). It also specifies how the text is to be organised (through both the goal decomposition and the use of RST relations, e.g., *context* and *elaboration*, in the figure, which explicitates the relationships between the nucleus and satellites). Operators include constraints which specify the conditions under which the operator is applicable. Finally, operators are of course written in a specific syntax (encoded as XML).

To be competent in writing discourse operators as shown in Figure 1, one needs to possess the following skills.

(1) computational linguistic skills: how to encode a discourse segment in terms of communicative goal and its decomposition (nucleus and satellites) – in particular, understanding of discourse theory, Rhetorical Structure Theory and discourse planning is desirable;
(2) domain knowledge: how to decide under which conditions a plan is applicable and where to get the data;
(3) writing skills: how to write a coherent document appropriate for their audience, and what is the functional role of different parts of the document;
(4) Understand the specific syntax.

Clearly, not many people possess all these skills. As a result, the plans are hard to write for most people, as they

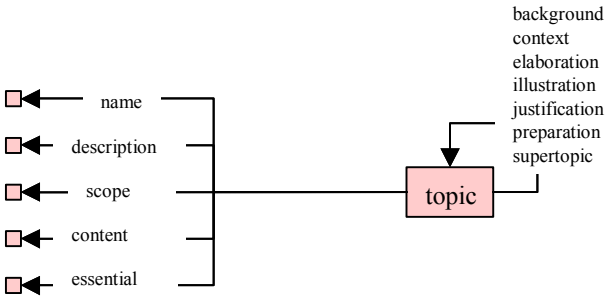

Figure 2. A schematic view of topic

require a lot of expertise, including technical, domain-oriented and writing expertise. Yet it is through plan operators that one specifies the types of text to be generated.

We would like people knowledgeable about the texts required in their domain and with writing skills (so people with writing skills in their domain) to be able to specify these texts, while keeping the advantages of the discourse planning approach, in particular keeping the discourse tree structure that enables further reasoning. To this end, we have started to design a new way to specify the plan operators, to decouple the specification of the structure of the text from the specification of how to retrieve the data, and to provide an abstract way to specify this structure – while still being able to produce the discourse tree.

Our approach is underpinned by three basic constructs: the content structure, the retrieval table and a set of generic operators. The content structure can be seen as a document definition model which is domain dependent. Therefore, it is to be authored by someone who knows how to write texts in their domain. The retrieval table is a registry of retrieval functions available in an application domain. It should be constructed by a software engineer in collaboration with a domain expert, as the data sources themselves are likely to be domain dependent. The set of generic operators is domain independent and have been authored while implementing the approach. They can now be used by different domain applications.

3.Modelling document for generation

We have introduced the notion of content structure. In a sense, the content structure is an abstract definition of a dynamic document. A content structure is composed of content nodes and relationships among them. Apart from hierarchical relationship, sibling nodes are related by RST [5]. The content structure can be seen as the blue-print for a tailored document. It (1) defines the rhetorical structure among different chunks of information in a dynamic document; (2) specifies relevant scopes for any particular topic/content node in relation to any contextual models; (3) links retrieval services to the content nodes so that the appropriate data can be retrieved from the underlying data sources.

With the constructs of content structure and retrieval services, we have devised a set of domain independent operators. These domain independent operators, or generic operators, can be used to operate on any content structure for any application domains to generate the desired domain dependent discourse trees. With this approach, there is no need for computational linguistists with domain knowledge to author conventional domain dependent discourse operators for a new application. Instead, only people with domain expertise and writing skills are required to author (domain dependent) content structure. In the following sub-sections, the constructs of content structure, retrieval table and generic operators will be further elaborated.

3.1.The Content structure

The content structure, a tree structure with content nodes, is a hierarchical representation of the document to be generated, which could be seen as a hierarchy of topics. At each level of decomposition, there are two types of content nodes (topics): essential and non-essential, essentially mirroring the nucleus/satellite distinction of RST: Essential nodes (nuclei) correspond to primary information that must be included in the text, while information contained in non-essential nodes (satellites) can be secondary or supportive. Again mirroring an RST structure, nodes have to be related with a rhetorical relation. In particular, non-essential nodes have to be related to essential nodes with an RST relationship, e.g., background, context, elaboration, justification, etc. Both types of nodes can be decomposed further.

Figure 2 shows a schematic view of a content node. Each node has a unique name, a textual description, a specification of its scope of applicability, its content proper and an attribute specifying whether it is essential or not.

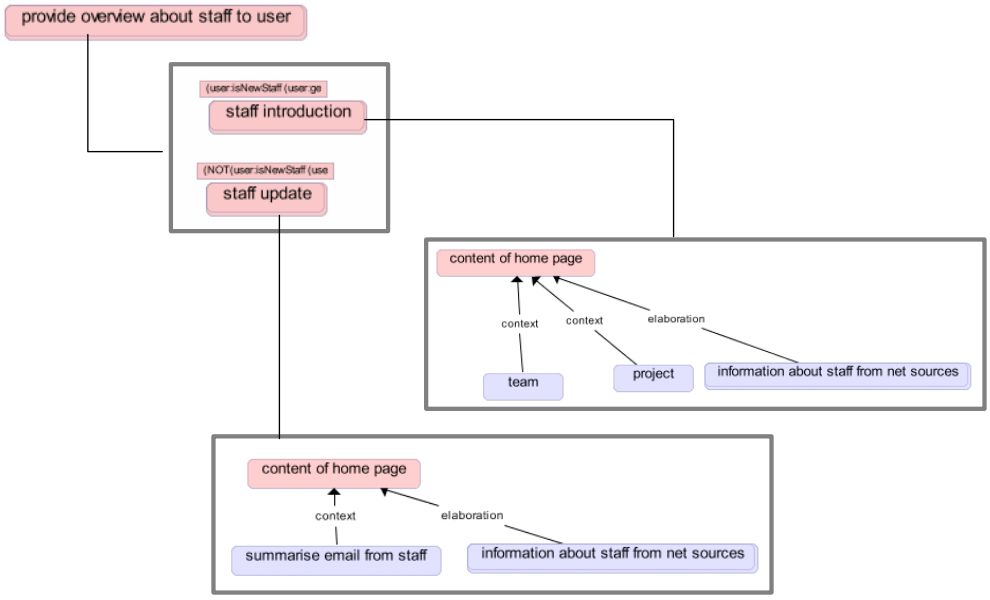

Figure 3. A content structure Fragment

Figure 3 shows a fragment of a content structure. This structure was defined for a new application we are constructing at present: StaffConnector. The system is concerned with providing information about a staff member. This is shown as the top-level node, shaded pink (or darker grey in black&white print), in the top left corner of the picture. The decomposition of the node is shown through the boxes. The author of this content structure decided that the virtual document to be generated in this case (as a web page) consists of either an introduction of the staff member, if the user (reader) is a new staff member him or herself, or an update on the staff member (consisting of a summary of the home page, a summary of email exchanges between this staff member and the user and information from internal sources). Both nodes are then decomposed further.

The scope attribute (shown in the rectangle above the node) defines the applicability or relevance of a topic under certain context. In this example, the "staff introduction" node has a user scope of "user:isNewStaff(user:getCurrentUser))". "user:getCurrentUser" will return the current user from the user model, and "user:isNewStaff(x)" is a Boolean function, which returns TRUE, if x is a new staff, exploiting internal human resource databases. As a result, the node "staff introduction" is only applicable to users who are deemed to be new staff. Scope for a topic is evaluated when the content structure is interpreted by the set of generic plan operators using the constraint mechanism of the plan operators, at runtime.

In essence, the scope provides a simple mechanism to enable "class" level customisation of tailored documents, i.e., inclusion or exclusion of content nodes based on contextual models (which refer to the user, the task, the domain, the discourse history or the environment [7]. (Instance level customisation would refer to different content delivered for different users, but corresponding to the same content node.)

Both nodes are further decomposed: "staff introduction" is further decomposed into the topics: "content of home page", "team" (the staff's position within the organisation hierarchy), "project" (the projects the staff is involved) and "staff on the net" (information related to the staff by searched from the net), where:

- "content of home page" node provides essential information (nucleus, indicated by the pink colour – or darker grey in B&W -- of the node);
- "team" and "project" nodes provide circumstantial information (satellites, shown in blue – or lighter grey in B&W, related to the nucleus by the RST relation called *context*, indicated on the link);
- "information from web sources" provides more information (a satellite, related to its nucleus with the RST relation called *elaboration*).

The content structure thus defines the abstract document structure and how different parts relate to each other. It also defines under what circumstances different parts (nodes) may be applicable (the scope). Finally, it also specifies how to acquire the actual content via *retrieval services* (not shown in the figure). Typically, retrieval services are mapped to the leaf nodes in the content structure. Retrieval services are registered into and managed by the retrieval table, explained in the next section.

As shown in Figure 4, each retrieval service is described by a set of attributes. The <id> field is a unique identifier and, by convention, describes the function of the retrieval service. The <service> field points to the software implementation that will obtain the appropriate data from designated data sources. The <description> field explains what the retrieval service is used for which is what the document designer will see the document design authoring tool. The data sources, which could be files, web sites, databases, and others, are specified in the <access> field. Each service is implemented individually for a specific retrieval purpose, and all retrieval services conform to a common protocol in the Myriad delivery platform.

```
<retrievalTable>
        <retrieval>
                <id>getProject</id>
                <service>GetProject</service>
                <format>xml</format>
                <description> get information about a project </description>
                <access>../../../resources/cmis_org.xml</access>
                <query></query>
                <load>true</load>
                <returnTime></returnTime>
                <returnSize></returnSize>
                <contenttype></contenttype>
                <stylesheet></stylesheet>
        </retrieval>
        <retrieval>
                <id>getHomepage</id>
                <service></service>
                <format>xml</format>
                <description>get staff's home page</description>
                <access></access>
                <query>(retrieval:GetPersonalAttribute (retrieval:getStaff)
                        homepage)</query>
                <load>true</load>
                <returnTime></returnTime>
                <returnSize></returnSize>
                <contenttype>text/html</contenttype>
                <stylesheet></stylesheet>
        </retrieval>
        … …
```

Figure 4. A fragment of retrieval table

3.2.Retrieval table

The retrieval table is used to manage retrieval services developed for specific applications. Retrieval services are software components which perform information retrieval functions. There are two types of retrieval services: elementary and composite. Elementary retrieval services directly retrieve needed information, while composite retrieval services are composed of elementary retrieval services or/and other composite retrieval services. Figure 4 shows a fragment of the retrieval table. Here, the first retrieval service, namely, "getProject", is elementary; and the second retrieval service ("getHomepage") is composite.

Composite retrieval services can be defined with the <query> field. In Figure 4, when the "getHomepage" composite retrieval service is called, the retrieval service "getStaff" will be evaluated first, returning the staff id. Then, the retrieval service "GetPersonalAttribute" will be evaluated.

The retrieval table thus contains both elementary and composite services and their definitions. Topics in the content structure refer to retrieval services in the retrieval table. When the content structure gets processed by the generic operators, the generation engine evaluates these retrieval services to acquire information.

```
… …
<operator>
<id>Present0</id>
<description>for composite topics</description>
<effect>(Present ?topic to ?user)</effect>
<constraint>(topic:hasslot ?topic essential )</constraint>
<constraint>(topic:hasslot ?topic normal )</constraint>
<constraint>(not(topic:hasslot ?topic scopeuser))</constraint>
<constraint>(mark name (topic:getslotfiller ?topic name))</constraint>
<nucleus>
  <value>(foreach ?esse (topic:getslotfillers ?topic essential)
       (Present ?esse to ?user))</value>
</nucleus>
<satellite>
  <type>optional</type>
  <relation>background</relation>
  <value>(foreach ?titl (topic:getslotfillers ?topic background)
       (Present ?titl to ?user))</value>
</satellite>
<satellite>
  <type>optional</type>
  <relation>background</relation>
   <value> (foreach ?cont (topic:getslotfillers ?topic context)
        (Present ?cont to ?user))</value>
 </satellite>

… …
<operator>
```

Figure 5. A fragment of a generic plan operator

3.3.Generic operators

Having the construct of content structure, we can process it with a set of generic operators and produce a discourse tree akin to the one produced through the conventional discourse operators. The generic operators, starting from the root node of the content structure, perform the following tasks:

(1) evaluate the scope of a node if there is one. The node will not be further processed if result of the evaluation is false;
(2) post appropriate discourse goals by branching out to children nodes;
(3) evaluate any retrieval services that may be attached to a content node, and bind the result in the appropriate structure of the discourse tree.

Figure 5 is an example of generic plan operator. It is worth noting that there is a limited set of generic discourse operators.

4. A test application

We tried this new approach on a new application we are developing: StaffConnector. We thus:

(1) developed the retrieval table and associated retrieval services;
(2) authored the domain dependent content structure;

For this application, we developed a simple user model to differentiate new staff from old staff. It is worth noting that the content structure and retrieval table development go hand in hand, even if they can be defined by different people. On the one hand, the content structure sets requirement for what retrieval services are needed. On the other hand, retrieval services determine the content that can be called from the content structure.

To facilitate the authoring of content structure, we have developed a content structure authoring tool, Constructor. This is what was illustrated in Figure 3.

The retrieval service "getHomepage" (Figure 4), which retrieves a specified staff's homepage from ICT Centre's intranet, is attached to the "staff home page" node. Retrieval services are also developed to acquire staff' team information, projects involved in, and information about a specified staff on the net (intranet, extranet, and internet).

Once the retrieval table and content structure are built, they are fed into the VDP together with the set of generic operators. Inside the VDP, the content structure is processed by the generic operators, where applicability scopes get assessed and retrieval services get executed. The outcome is a fully fledged discourse tree, which is then processed with presentation operators. The final outcome is a set of HTML documents which are tailored to the user model. Figure 6 shows the main page of the staff overview application. As it can be seen, there are two panes in the window. The left pane shows the table content view of the document. It provides a global view of the document and can be used to navigate to different sections within the document. By default, the first section is displayed in the right pane. (The layout and presentation are performed by another stage in the planning process.)

Figure 6. The main page for the staff overview application

5.Discussion and future work

One of the challenges in developing planning based tailored document generation systems is the issue of authoring discourse operators. The difficulty lies in the requirement of several kinds of expertise simultaneously. To be a competent discourse operator author, one needs to possess knowledge about writing, the application domain and computational linguistics. Consequently, few people are qualified to be able to write discourse operators.

In this paper, we have presented a novel approach to specify the plan operators, decoupling the specification of the structure of the text from the specification of how to retrieve the data, and providing an abstract way to specify this structure – and still being able to produce the discourse tree that is desirable to perform a number of reasoning tasks after the content planning stage. In doing so, we intend to enable people knowledgeable about the texts required in their domain to be able to specify these texts, while keeping the advantages of the discourse planning approach, in particular keeping the discourse tree structure that enables further reasoning. This approach is underpinned by three major constructs, namely, the content structure, the retrieval table and the generic operators.

The content structure is a document definition model which needs to be constructed for every new application. The retrieval table defines retrieval functions for acquiring information from various data sources. Having introduced the content structure, we have developed a finite set of generic operators. With theses generic operators, discourse tree can be generated from any domain dependent content structures. In effect, the issue of discourse operator authoring is transformed into the issue of content structure authoring.

To facilitate the task of content structure authoring, we have built a content structure authoring tool, the Constructor. While we have demonstrated our approach by a simple example, we have also discovered some limitations with our current design. One of the limitations is that the abstract construct of iteration can not be handled. However, we believe that that limitation can be overcome by extending our current modelling constructs. That is what we would like to look into in our next step. Furthermore, we would like to extend our approach to such an extent that it would comfortably handle all possible cases in discourse trees. Another item high on our agenda is to evaluate the usability of our approach.

Acknowledgements We wish to thank other members of the group, in particular Andrew Lampert and Akshay Bhurtun.

References

[1] M.K. Campbell, B.M. DeVellis, V.J. Strecher, A.S. Ammerman, R.F. DeVellis, and R.S. Sandler, (1994). *Improving dietary behavior: The effectiveness of tailored messages in primary care settings*. American Journal of Public Health, 84:783–787.

[2] N. Colineau, C. Paris and M. Wu (2004). *Actionable Information Delivery*. In Revue d'Intelligence Artificielle (RSTI – RIA), Special Issue on Tailored Information Delivery, 18(4), 549-576.

[3] N. Colineau and S. Wan (2001). *Mobile delivery of customised information using Natural Language Generation*. In Monitor (Special Issue on Wireless

Communication Special), 26(3), September-November 2001, 27-31.

[4] A. Edmunds and A. Morris. *The problem of information overload in business organisations: A review of the literature*. International Journal of Information Management, 20(1):17–28, February 2000.

[5] W.C. Mann and S.A. Thompson *"Rhetorical Structure Theory: Toward a functional theory of text organisation"*, In Text 8 (3), 1988, pp. 243-281.

[6] J. Moore and C. Paris (1993) *Planning Text for Advisory Dialogues: Capturing Intentional and Rhetorical Information*. In Journal of Computational Linguistics; 19 (4), December 1993. pp 651 - 694.

[7] C. Paris, M. Wu, K. Vander Linden, M. Post and S. Lu (2004). *Myriad: An Architecture for Contextualized Information Retrieval and Delivery* (2004). In AH2004: International Conference on Adaptive Hypermedia and Adaptive Web-based Systems. August 23-26 2004, The Netherlands. pp.205-214.

[8] C. Paris, M. Wu, A-M Vercoustre, S. Wan, P. Wilkins and R. Wilkinson (2003). *An Empirical Study of the Effect of Coherent and Tailored Document Delivery as an Interface to Organizational Websites*. In The Proceedings of the Adaptive Hypermedia Workshop at the 2003 User Modelling Conference, Pittsburgh, USA, June 22, 2003. pp 133 - 144.

[9] Ehud Reiter. 1995. *NLG vs. templates*. In Proceedings of the 5th European Workshop on Natural Language Generation, Leiden, The Netherlands.

[10] E Reiter and R Dale (2000) *Building Natural Language Generation Systems*. Cambridge University Press.

[11] C.S. Skinner, V.J. Strecher, and H. Hospers, (1994*). Physicians' recommendations for mammography:Do tailored messages make a difference?* American Journal of Public Health, 84:43–49.

[12] V.J. Strecher, M., Kreuter, D.-J. Den Boer, S. Kobrin, H.J. Hospers, and C.S. Skinner. (1994). *The effects of computer-tailored smoking cessation messages in family practice settings.* The Journal of Family Practice, 39:262–270.