Case-Based Reasoning for Knowledge-Intensive Template Selection During Text Generation

Raquel Hervás and Pablo Gervás

Departamento de Sistemas Informáticos y Programación
Universidad Complutense de Madrid, Spain
raquelhb@fdi.ucm.es,pgervas@sip.ucm.es

Abstract. The present paper describes a case-based reasoning solution for solving the task of selecting adequate templates for realizing messages describing actions in a given domain. This solution involves the construction of a case base from a corpus of example texts, using information from WordNet to group related verbs together. A case retrieval net is used as a memory model. A taxonomy of the concepts involved in the texts is used to compute similarity between concepts. The set of data to be converted into text acts as a query to the system. The process of solving a given query may involve several retrieval processes - to obtain a set of cases that together constitute a good solution for transcribing the data in the query as text messages - and a process of knowledge-intensive adaptation which resorts to a knowledge base to identify appropriate substitutions and completions for the concepts that appear in the cases, using the query as a source. We describe this case-based solution, and we present examples of how it solves the task of selecting an appropriate set of templates to render a given set of data as text.

1 Introduction

Template-based solutions for natural language generation rely on reusing fragments of text extracted from typical texts in a given domain, having applied to them a process which identifies the part of them which is common to all uses, and leaving certain gaps to be filled with details corresponding to a new use. For instance, when conveying the information that John moved to Atlanta, a template such as _ *moved to* _ may be used, filling in the gap with appropriate strings for John and Atlanta. Whereas more complex natural language generation systems based on the use of grammars can have rich stages devoted to selecting fresh combinations of words to convey the same meaning, template-based systems are faced with an additional difficulty. The fact that templates are made up of words that are not accessible to the system makes the system blind to their appropriateness as means of conveying a given idea for a specific set of arguments. Annotating the templates with tags that indicate the circumstances under which it is appropriate to use the template would solve the problem, but it eliminates some of the advantages of the template solution over more knowledge-rich approaches.

Applying a case-based solution presents the advantage that the information needed to solve the problem can be obtained from the original examples of appropriate use that gave rise to the templates. By associating a case with each template, with case attributes consisting of conceptual descriptions of the arguments that were used for the template in the original instance, a case-based reasoning solution can be employed to select the best template for realizing a particular message. This approach has a certain psychological plausibility. People do not always create new linguistic constructions each time they need to express an idea not used before, but rather they appeal to their memory of expressions they have used or heard in the past looking for the best way to express the new idea. They remember other situations where they have expressed similar ideas, and the phrasing they used in each situation. In this process they take into account existing relations between the elements of the lexicon they already know.

The present paper describes a case-based solution for the task of selecting adequate templates for realizing messages describing actions in a given domain. The goal is to achieve coverage of a broad range of messages by combining instances of a restricted set of templates, providing automated means for dealing with overlaps between the information conveyed by the templates found, and to ensure coherent use of context information - in the shape of a knowledge base for the domain accepted as input - whenever the resulting templates need to mention information that was not explicit in the given query.

Section 2 presents a review of previous work in the relevant fields. Section 3 describes a case-based solution to template selection, outlining the inputs to consider, the construction of the case base, and the main CBR processes involved. Section 4 describes in detail an example of system operation, and section 5 presents conclusions and further work.

2 Lexicalisation and Knowledge Intensive CBR

The general process of text generation takes place in several stages, during which the conceptual input is progressively refined by adding information that will shape the final text [1]. During the initial stages the concepts and messages that will appear in the final content are decided (*content determination*), these messages are organised into a specific order and structure (*discourse planning*), and particular ways of describing each concept where it appears in the discourse plan are selected (*referring expresion generation*). This results in a version of the discourse plan where the contents, the structure of the discourse, and the level of detail of each concept are already fixed. The *lexicalization* stage that follows decides which specific words and phrases should be chosen to express the domain concepts and relations which appear in the messages. A final stage of *surface realization* assembles all the relevant pieces into linguistically and typographically correct text.

The most common model of lexicalisation is one where the lexicalisation module converts an input built from domain concepts and relations organised as a graph into an output built from words and syntactic relations also organised as a graph. Cahill [2] differentiates between "lexicalization" and "lexical choice". The first term is taken to indicate a broader meaning of the conversion of something to lexical items, while the second is used in a narrower sense to mean deciding between lexical alternatives representing the same propositional content. Stede [3] proposes a more flexible way of attaching lexical items to configurations of concepts and roles, using a lexical option finder that determines the set of content words that cover pieces of the message to be expressed. These items may vary in semantic specificity and in connotation, also including synonyms and nearsynonyms. From this set, the subsequent steps of the generation process can select the most preferred subset for expressing the message.

In template-based generation, the selection of templates is part lexicalization and part surface realization, in the sense that it determines some of the words that appear in the final text, but it also defines how they are put together into a (hopefully) correct linguistic statement. For this reason, template selection inherits a distinction equivalent to that pointed out by Cahill: one can talk about *rigid template assignment* - where a given type of message is always realised by the same template -, and *template choice* - where a given type of message can be realised by several templates and mechanisms must be provided for deciding when to use each possible template. A solution similar to that proposed by Stede for pure lexicalization would be a good way of implementing template choice.

Case based approaches have been applied to natural language processing (NLP) problems in the past. These natural language solutions process a text by retrieving stored examples that describe how similar texts were handled in the past. Examples of particular applications are stress acquisition [4], word sense disambiguation [5] and concept extraction [6]. A good review of applications of machine learning techniques in general to NLP tasks can be found in [7].

Knowledge Intensive Case-Based Reasoning (KI-CBR) relies on taxonomical information about the concepts handled by a CBR system in order to improve the results of the processes involved. This taxonomical information usually takes the form of specific ontologies of domain information, sometimes coupled with generic ontologies about CBR concepts [8]. A typical use involves resorting to the taxonomy for computing similarity between cases.

When building knowledge resources for supporting this type of CBR, accepted practice recommends the reuse of previous existing ones. The WordNet lexical database [9] has been widely used for knowledge-based systems, including appplications to CBR in domains such as design support [10].

Case Retrieval Nets (CRNs) [11] are a memory model developed to improve the efficiency of the retrieval tasks of the CBR cycle. They are based on the idea that humans are able to solve problems without performing an intensive search process, but they often start from the given description, consider the neighbourhood, and extend the scope of considered objects if required.

The basic item in the context of the CRNs are so-called Information Entities (IEs). These represent any basic knowledge item in the form of an attribute-value pair. A case then consist of a set of such IEs, and the case base is a net with nodes for the entities observed in the domain and additional nodes denoting the particular cases. IE nodes may be connected by similarity arcs, and a case node is reachable from its constituting IE nodes via relevance arcs. Different degrees of similarity and relevance are expressed by varying arc weights. Given this structure, case retrieval is carried out by activating the IEs given in the query case, propagating this activation according to similarity through the net of IE nodes, and collecting the achieved activation in the associated case nodes.

Case retrieval nets have been used for lexicalization before. Hervás and Gervás [12] presented an application of a CRN to heuristic lexicalization in the context of a NLG application. The case base employed was obtained from the set of formalised documents used as input to the generator. The experimental results showed that the use of the case-based reasoning paradigm for the task of lexicalisation is a good approximation whenever enough information is available in the case base to express in an acceptable form any new request. If queries beyond the scope of the input were tried, the system performed poorly. It remained an open question whether the use of a larger case base with broader coverage would improve results.

3 Case Based Solutions for Template Selection

Lexicalization based on templates selects words from the vocabulary to describe the concepts involved in the current draft, using lexical tags for static concepts, as characters and scenarios, and templates for actions and verbs, providing structure to the sentences. Templates partly solve the need for having an explicit grammar, but the knowledge base provides the required information to solve issues like number and gender agreement. This is an acceptable method when operating in restricted domains, but results can be poor if complex actions have to be expressed. Complex actions require the introduction of lexical chains that are employed exclusively for a specific verb in some context. This introduces an unwanted rigidity in the system, because it makes the task of extending the vocabulary an arduous one. This solution also implies that the vocabulary holds no semantic information about actors or objects involved in an action.

As an alternative, we have implemented a case-based lexicalisation module. When human beings talk or write, they do not always invent entirely new sentences whenever they need to express a specific idea that they have never before put into words. Instead, sometimes they search for relations between the new idea to be expressed and other ideas expressed previously, taking the same vocabulary and adapting it as required. They reuse previous experience to solve a new case.

This module relies on subsequent processing of its output by an accompanying surface realization module. This module is in charge of putting together the selected terms and templates. Additionally, it carries out a basic orthographic transformation of the resulting sentences. Templates are converted into strings formatted in accordance to the orthographic rules of English - sentence initial letters are capitalized, and a period is added at the end.

The specific architecture of the NLG application that uses these modules is implemented using cFROGS [13], a framework-like library of architectural classes intended to facilitate the development of NLG applications. It is designed to provide the necessary infrastructure for developing NLG applications, minimizing the implementation effort by means of schemas and generic architectural structures commonly used in these systems.

3.1 Inputs to the NLG module

Three particular inputs to the NLG module are relevant to the work described here: the knowledge base, the vocabulary and the discourse plan.

The knowledge base The knowledge base contains the relevant conceptual information about the domain, in such a way that the generator can consult it and use it. It is organized as a tree, including individuals, locations, objects, relations between them and their attributes. The facts in the knowledge base are domain concepts that are used to instantiate the templates when representing cases. The concepts appearing in the cases are organized as a taxonomy needed to compute their similarities. Each type of concept is divided into several subtypes.

The vocabulary The vocabulary contains all the lexical information essential to write the final text. It is structured as a tree as well, very similar to the knowledge base one, with the difference that each fact or relation has a lexical tag associated to its eventual realization in the final text.

In the vocabulary that the system uses, a lexical tag made up of one or more words is assigned to each concept in the domain. This is used for lexicalising individual concepts, with little choice given. The vocabulary for actions or verbs becomes more complex: it is stored in the form of cases, where each case stores not only the corresponding template but also additional information concerning the type of case, the elements involved in the action, and the role that those elements play in the action. The types of actions that appear in this module are shown in Table 1.

It is important to take into account that the structure of each one of these types is not rigid. They will not always have the same elements, nor in the same order. A clear example is provided by the verbs ‘leave’ and ‘go’, both of type `Move`. The first one has an attribute `From` to indicate where the character is coming from, whereas the second one has an attribute `To` that indicates his destination.

Verbs are particularly important in the present context because we are considering narrative tales which have little descriptive depth. This implies that verbs carry a significant part of the communication effort.

A case is not an abstract instance of a verb or action, but rather a concrete instance in which specific characters, places and objects appear. These elements are stored in the module’s knowledge base. This allows the establishment of relations between them when it comes to retrieving or reusing the cases.

Table 1. Types of actions

Type	Characteristics of action
Move	a character or an object changes of location
Atrans	possession of a character or an object is transferred
Fight	physical confrontations between characters
Ingest	a character ingests something (either another character or an object)
Propel	a physical force is applied to an object
State	change of state of a character or an object
Use	an element of the domain participates in the action
Feel	involve feelings, both positive or negative
Speak	a character or an object express an idea out loud

Not all the attributes in a case act later as slot-fillers in the corresponding template. Some attributes - like the specification of initial and final states in a state change - are not explicitly mentioned in a template. For example, the case for *kill* indicates an initial state `alive` and a final state `dead`, but these are not mentioned in the surface form of the template. Such attributes appear enclosed in culry brackets in the representation of the case.

Examples of cases for the different types of action are given below. The associated templates are shown below for each case:

```
TYPE:   LEX:    ACTOR:  OBJECT:
FIGHT   attack  witch   Hansel
```

_ *attacked* _

```
TYPE:   LEX:    ACTOR:       OBJECT:     {FEELING:}
FEEL    envy    stepsisters  Cinderella  bad
```

_ *envied* _

```
TYPE:   LEX:    ACTOR1: ACTOR2:   {INI:}  {FINAL:}
STATE   marry   knight  princess  single  married
```

_ *married* _

The discourse plan The discourse plan is the structure of the information that is to be rendered as text. Each line of this input corresponds to a paragraph sized portion of the text, containing information about a sequence of actions, the place where they take place, the characters involved, and the objects used in them. The solution presented here involves only a particular line of the discourse plan. The rest of the discourse plan would have to be treated in the same way one line at a time. For some complex decisions involved in case adaptation - discussed in section 3.5 - the context as featured in the discourse plan may need to be consulted.

3.2 Building The Case Base

To ensure a broad coverage of possible inputs, the case base has been built by combining two sources: WordNet and a corpus of texts selected as typical examples of the type of text desired as output. In order to provide a broad enough choice of templates - in the sense described above - a set of possible templates must be assigned to each type of action. WordNet is used as a basic source to obtain a set of possible verbs for conveying a given type of action. This set of verbs must be filtered to ensure that the final selection of verbs conforms with typical usage in the desired genre. A corpus of texts is used to filter out the WordNet senses that are inappropriate for the genre of the desired outputs. For the selected senses, the corpus provides examples of use, which are used to generate the cases.

The corpus employed consists of 109 classic fairy tales, which included a total of 9852 sentences. These have been obtained from Internet web sites presenting collections of fairy tales in English, and they include a collection of Aesop's fables, a selection of Afanasiev's collection of Russian fairy tales, and tales from the selections by Andersen, the Grim brothers, and Perrault.

For each case, the concepts appearing in the example found in the corpus - which become information entities in the case retrieval net used to store the cases - must also be inserted into the knowledge base. To ensure appropriate performance, they must be inserted at the correct place in the taxonomy that organises the knowledge base. This is important because the system uses the relative positions in this taxonomy to calculate similarity between the query and the cases during retrieval.

The general process of constructing a particular case is better illustrated by means of an example.

The MOVE type of action is associated with the concept 'move' found in WordNet. Sixteen senses are provided for it. Of those, the first three are relevant in this situation: sense 1 (travel, go, move, locomote), sense 2 (move, displace) and sense 3 (move so as to change position, perform a nontranslational motion). For each one a number of possible words for that sense are listed (132 for sense 1, 90 for sense 2 and 99 for sense 3). These must be filtered with respect to the corpus. For instance, 51 of the possible words for sense 1 do not appear at all in the corpus.

One of the possible words suggested by WordNet as a sense of 'move' is:

```
travel -- (undergo transportation as in a vehicle; "We travelled
North on Rte. 508")
```

The corpus provides the following examples of use of that word:

One day the king went travelling to distant lands
Then we [Simbad (and the merchants)] traveled many days across high mountains until we came to the sea, where we set sail.

These examples give rise to the corresponding cases:

```
TYPE:   LEX:       ACTOR:  TO:
MOVE    travel-to  king    distant-lands

TYPE:   LEX:           ACTOR:  ACROSS:
MOVE    travel-across  Simbad  mountains
```

The nature of the documents used to build the corpus - children's fairy tales - presents the advantage that the range of concepts involved, both in terms of verbs and nouns, is limited.This results in a certain degree of lexical redundancy which simplifies the knowledge acquisition process. This process is carried out semi-automatically. A dependency analysis is carried out for the sentences in the corpus using Minipar [14]. Verbs and the nouns that depend on them are identified. For each verb, and action must be built and a template must be generated. A type is assigned to each action. This type constitutes a strong restriction during retrieval, so it should be used with caution. As discusses in section 4, maximal flexibility in the use of the system is obtained by ommiting this attribute during retrieval. The concepts corresponding to the nouns identified in the corpus must be inserted into the knowledge base. Both action construction and noun concept insertion require human supervision., However, the fact that nouns and verbs are treated asymetrically reduces the actual effort involved in processing a large corpus: no composite structure is built for representing nouns, and the insertion of verbs into the knowledge base is not subject to positional restrictions.

3.3 The Case Base

Cases are stored in a Case Retrieval Net. This model is appropriate for the problem under consideration, because on one hand our cases consist of attribute-value pairs that are related with one another, and on the other hand the queries posed to the module will not always be complete. To find a lexical tag for a given action, the CRN is queried with the class of elements involved in the action.

The vocabulary of the module is built from the case base. For each attribute-value pair in the cases an information entity is created. For each case, a node is created which holds references to the information entities that are contained. When introducing an IE, if that entity has already appeared in another case it is not duplicated. Instead, another association is created between the new case and the existing information entity.

As IEs are inserted to form the net, it is necessary to establish a measure of similarity between them. This is done by reference to the module's knowledge base, in which the different concepts of the domain are organised into a taxonomy. The similarity between two entities is calculated by taking into account the distance between them in the knowledge base and using Formula 1. H is the maximum height in the knowledge base.

$$sim(c1, c2) = 1 - (1 + distance(c1, c2))/(H * 2) \tag{1}$$

The distance between two concepts is calculated by finding their first shared ancestor, and adding up the distance between this ancestor and each of the concepts. It can be seen as the number of nodes we have to pass when going from one of the concepts to the other. It is also necessary to have a similarity value for each entity with itself. This value is always 1, the maximum possible.

Each of the IEs is related to the cases to which it belongs with a certain value of relevance. In the implemented module, the maximum relevance within a case corresponds to the attribute `Type` with value 1, and the rest of the elements have relevance 0.5. This is because when retrieving cases we are mainly interested in the type of action that we are looking for, rather than which elements are involved in it. However, it can occur that the module retrieves a case of a different type, if the similarity weights of the attributes of the case are high enough.

3.4 Case Retrieval

The retrieval task starts with a partial or complete problem description, and ends when a matching previous case has been found. In our module, the retrieval of cases is directly handled by the Case Retrieval Net and its method of similarity propagation. Starting from a partial description of the action we need to lexicalise, the retrieval of the more similar cases is done by calculating an activation value for each case in the case base. The ones with higher activation are the more similar ones to the given query. This calculation is performed in three steps:

1. The IE nodes that correspond to the query are activated. If they are not in the net because they did not belong to any case in the case base, the corresponding nodes are inserted at the time of querying, calculating the similarity and relevance weights using the knowledge base. The nodes corresponding to the query are assigned an activation value of 1, and the rest a value of 0.
2. The activation is propagated according to the similarity values of the arcs. This is performed by looking over all the entity nodes of the net and by calculating for each one its activation value using its own activation and its similarity with the rest of IE nodes. This is achieved by using Formula 2 (where N is the total number of IE nodes).

$$activation(e) = \sum_{i=1}^{N} (sim(e_i, e) * activation(e_i)) \tag{2}$$

3. The achieved activations in the previous step are collected in the associated case nodes, calculating the final activations of the cases also considering the relevance weights of the arcs that connect the cases with their entities. This final activation value of the cases is calculated with Formula 3.

$$activation(c) = \sum_{i=1}^{N} (rel(e_i, c) * activation(e_i)) \tag{3}$$

Once we have the final activation in the cases, the one with the higher value is returned by the net. It would be possible to take not only the most similar one, but a set with the most similar cases to the query.

3.5 Case Reuse

Each retrieved case has an associated template from the vocabulary for the verb or action it represents. In the process of reusing the case we have obtained from the net, we have to substitute the attribute values of the past case with the query values. Here we have three different possibilities:

1. The retrieved case and the query have the same set of attributes.
2. The query has more attributes than the retrieved case.
3. There are more attributes in the retrieved case than in the query.

For situation 1, the values of the retrieved case are simply replaced with the values in the query. The template corresponding to the retrieved case is filled with the new values. At the end of the reuse process the query has been assigned a correct template to realize as text the message it conveys.

For situation 2, the attributes of the retrieved case are filled with the corresponding query values. The resulting adaptation is a partial solution to the problem posed by the query. A secondary retrieval process is set in motion, using as a query simply the set of attributes in the query that could not be accommodated in the partial solution provided by the first case retrieved by the system. This query includes no specific type of action, and it relies on the ability of the case retrieval net for case completion to provide a case with a type of action that matches the given arguments.

For situation 3, there will be vacant attributes in the corresponding solution. The easiest solution is to keep the values of the past case in the slots for which the query does not specify any value. Better results can be obtained by consulting the system knowledge base for concepts that the knowledge base shows as related to those appearing in the query. In order to be appropriate as fillers for the vacant slots, these concepts must be within a given threshold of similarity - in terms of relative distance within the taxonomy - with respect to the original values given in the retrieved case for those attributes. In situations where lexicalization of a particular message forms part of a larger context - such as a larger text - better results are obtained by searching the neighbouring messages in the discourse.

3.6 Case Revision and Retainment

A very complex set of linguistic, cognitive and pragmatic constraints must be taken into account when validating any natural language solutions generated in this manner. The contribution of an expert in the domain is required to revise the results achieved by the module, and no automated solution to this stage of the CBR cycle is contemplated so far.

4 An Example of System Operation

To show how the system operates, an example is presented. Suppose the system is presented with a query such as:

```
ACTOR: prince, OBJECT: dragon, WITH: sword, INI: alive, FINAL: dead
```

The text that would correspond to this query would presumably be *"The prince killed the dragon with a sword"*.

The retrieval process results in the following case:

```
TYPE:   LEX:    ACTOR:  OBJECT: WHERE:  {INI:}  {FINAL:}
STATE   kill    peasant snake   forest  {alive} {dead}
```

This case is chosen because the values for the attributes `INI` and `FINAL` are equal, and the similarities between the other concepts, computed using the taxonomy, is high (*'prince'* and *'peasant'* are immediate siblings in the taxonomy - descendants of *'person'* - and *'dragon'* and *'snake'* are descendants of nodes that are immediate siblings - *'flying-creature'* and *'non-flying-creature'*).

The case obtained during the retrieval process contains attributes (`TYPE` and `WHERE`) for which no values are given in the query. The `TYPE` attribute is special and it will be simply inherited by the contribution obtained from this case for the final solution. The absence of TYPE in the query is intentional. The final result provided by the system may have to be built up from several cases of different types in order to account for all the attributes given explicitly in the query. To include an explicit type in the query would restrict the set of possible actions that can be included in the final result to those matching the explicit type. This would defeat the purpose of the technology we are using. The presence of unfulfilled attributes of other kinds - such as the `WHERE` attribute in this example - triggers a secondary process of searching the knowledge base for possible values for those attributes. Elements in the knowledge base related with those elements appearing in the query are considered as possible candidates to fill the additional attribute slots in the second retrieved case. The most similar ones - according to relative proximity within the taxonomy - are considered. In this instance, the knowledge base is queried for elements related to `prince`, `dragon` or `sword` which are similar to `forest`. The relations for these three concepts are shown in the paraphrase of the knowledge base[1] given in Table 2.

The system returns `cave` because the knowledge base contains information about the dragon living in a cave, and `cave` is similar to `forest` in the taxonomy. Other choices would have been `palace` or `princess`, related in the knowledge base to the `prince`, but their calculated similarities to `forest` are lower.

Since there are attributes present in the query for which no slot is available in the retrieved case (`WITH`), a second retrieval process is triggered with the following query, resulting from a selective restriction of the original query to those

[1] Relations in the actual knowledge base are represented in terms of instance identifiers, which would be meaningless for readers in this context.

Table 2. Relations in the knowledge base

```
relations:
   relation(prince,palace,live)
   relation(prince,sword,have)
   relation(prince,princess,love)
   [...]
   relation(dragon,cave,live)
   [...]
```

attributes not provided in the case retrieved in the first instance (the subject and object of the action are retained, to ensure soundness of the result):

```
ACTOR: prince, OBJECT: dragon, WITH: sword
```

This second retrieval process returns the following case:

```
TYPE:    LEX:     ACTOR:   OBJECT:  WITH:
FIGHT    attack   hunter   lion     spear
```

The final result of the complete process is an adaptation of the set of retrieved cases in all the required retrieval process, together with an assignment of values to their attributes, either from the original attributes in the query or from values related to them obtained from the knowledge base. The fact that the query had no explicit TYPE attribute has permitted that the solution be composed of several instances with different types.

The associated templates are shown below for each case.

```
TYPE:    LEX:     ACTOR:   OBJECT:  WHERE:   {INI:}   {FINAL:}
STATE    kill     prince   dragon   cave     {alive}  {dead}
```

_ *killed* _ *in* _

```
TYPE:    LEX:     ACTOR:   OBJECT:  WITH:
FIGHT    attack   prince   dragon   sword
```

_ *attacked* _ *with* _

To improve readability, any attribute slots whose values have already been mentioned in preceding cases within the same response are marked, to indicate that subsequent stages of the generation process should render them as pronouns.

After surface realization, the result provided by the system for the original query would be:

The prince killed the dragon in the cave. He attacked it with a sword.

This result is not exactly what we were looking for, but it conveys all the desired information. The vocabulary does not have the exact template needed in this case, but the system combines the templates and knowledge base resources it possesses to compose an alternative phrasing for the requested message.

5 Conclusions and Future Work

The case-based solution described in this paper presents the advantage of achieving coverage of a broad range of messages by combining instances of a restricted set of templates, providing automated means for dealing with overlaps between the information conveyed by the templates found, and ensuring coherent use of context information - in the shape of a knowledge base for the domain accepted as input - whenever the resulting templates need to mention information that was not explicit in the given query. By resorting recursively to processes of case retrieval with progressively reduced versions of the query till all the data in the query have been covered by at least one case, the system automatically obtains the best set of cases that cover the data with minimal overlap. Whenever the selected cases involve information that was not explicitly available in the query, the use of the input knowledge base guarantees that any additional information drafted into the final result is coherent with the particular set of data under consideration.

The main advantage of this method with respect to other template selection approaches is that the system does not need an exhaustive set of templates, as CRNs work by approximation and can retrieve similar cases for unknown queries due to the automatic semantic relations attained in the net. A classic problem in natural language generation is the “generation gap” described by Meteer [15], a discrepancy between what can be expressed in the text plan and what the particular realization solution can actually convert into text. In terms of templates, the “generation gap” occurs when the input calls for messages not explicitly contemplated in the set of templates in use. The present system ensures that such messages can be conveyed by a combination of simpler templates, adequately linked together by occurrences of coreferring elements. CRNs can handle partially specified queries without loss of efficiency, in contrast to most case retrieval techniques that have problems with partial descriptions. Given only a part of a case, the net can complete the rest of its content. This behaviour is similar to that suggested by Stede [3] for the lexicalization task.

Cases need not be described by attribute vectors. There are features that would be relevant for some cases but not for the others. This allows for cases associated to different templates of the same type to have different number of attributes, which ensures easy treatment of sentences with a wide choice of complements - for instance *“I have lunch”*, *“I have lunch in the office”*, *“I have lunch at my desk”*, *“I have lunch at midday”*.

Insertion of new cases (even with new attributes) can be performed incrementally by injecting new nodes and arcs. This is a particular advantage since any extension of the corpus may lead to the addition of new cases.

The difficulties presented by the knowledge acquisition have been partially addressed by the semi-automatization of the analysis of the corpus. We are currently working on improving this aspect of the system. Although it will probably be impossible to fully automate the process of acquisition, the method presented here represents a significant improvement on manual approaches to the development of template-based generators. Both approaches require the construction of the templates and the representation of the concepts. However, whereas an altogether manual approaches requires the additional elaboration of explicit criteria to guide the correct use of templates, the approach presented here provides an automatic case-based decision process for template selection.

The approach employed in this paper for actions may be extended to other elements in a story, such as characters, objects, locations,... This would require a specific notation in which these elements are described as a collection of attribute-value pairs. We have chosen to focus on actions until we have explored the potential of the technique. The exploration of such extensions to other elements will be contemplated as further work. However, the possible effect upon the complexity of knowledge acquisition must be considered.

The representation of actions in the current version of the system is very simple. The resulting texts would improve significantly if a more complex set of actions where considered. Template-based generators have obtained results comparable to more elaborate solutions by resorting to recursive use of templates [16]. In our approach, this would correspond to allowing actions to be represented as nested cases, where a case would be constructed not only of attribute-value pairs, but also attribute-case pairs, where the value for some attribute may itself be a complete case - with an associated template. Recursive nesting of cases would allow recursive use of templates. The retrieval and adaptation stages would have to be adapted to deal with this recursive nature.

In order to tackle the complexity arising from this enhancement, we contemplate two possible sources of inspiration. One is to consider the use of primitives to build complex actions from simple ingredients, along the lines of Schank's Conceptual Dependency theory [17]. Schank proposed an open set of primitives to express the meaning of any sentence in terms of primitives, using a complex system for representing states and relationships. Another is to define a complex conceptual taxonomy of actions, relying for their manipulation on the same techniques employed for handling individual concepts in the current system. This would allow a homogeneous treatment of knowledge through the system, and may lead to easier interactions between the different types of knowledge. To organise such a taxonomy of actions, WordNet would be a valuable source.

Efficiency issues have not been contemplated so far, but as the size of the case base rises, they are becoming relevant. A possible way of reducing this risk would be to implement *lazy spreading activation* [18] in the Case Retrieval Net. Instead of propagating activation to all entity nodes, and then to all case nodes, propagation takes place progressively from most similar nodes to not so similar nodes. Once enough case nodes have been activated to reply to the query, propagation stops.

References

1. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press (2000)
2. Cahill, L.: Lexicalisation in applied NLG systems. Technical Report ITRI-99-04 (1998)
3. Stede, M.: Lexical options in multilingual generation from a knowledge base. In Adorni, G., Zock, M., eds.: Trends in natural language generation: an artificial intelligence perspective. Number 1036. Springer-Verlag (1996) 222–237
4. Daelemans, W., Gillis, S., Durieux, G.: The acquisition of stress: a data-oriented approach. Comput. Linguist. **20** (1994) 421–451
5. Ng, H.T., Lee, H.B.: Integrating multiple knowledge sources to disambiguate word sense: an exemplar-based approach. In: Proceedings of ACL 1996, NJ, USA, ACL (1996) 40–47
6. Cardie, C.: Integrating Case-Based Learning and Cognitive Biases for Machine Learning of Natural Language. Journal of Experimental and Theoretical Artificial Intelligence **11** (1999) 297–337
7. Daelemans, W.: Introduction to the special issue on memory-based language processing. J. Exp. Theor. Artif. Intell. **11** (1999) 287–296
8. Gervás, P., Díaz-Agudo, B., Peinado, F., Hervás, R.: Story Plot Generation based on CBR. Journal of Knowledge-Based Systems **18** (2005) 235–242
9. Miller, G.A.: Wordnet: a lexical database for English. Commun. ACM **38** (1995) 39–41
10. Gomes, P., Pereira, F.C., Paiva, P., Seco, N., Carreiro, P., Ferreira, J., Bento, C.: Selection and Reuse of Software Design Patterns Using CBR and WordNet. In: Proc. of the 15th International Conference on Software Engineering and Knowledge Engineering (SEKE'03). (2003)
11. Lenz, M., Burkhard, H.D.: Case Retrieval Nets: Basic Ideas and Extensions. In: KI - Kunstliche Intelligenz. (1996) 227–239
12. Hervás, R., Gervás, P.: Case Retrieval Nets for Heuristic Lexicalization in Natural Language Generation. In Cardoso, A., Bento, C., Dias, G., eds.: Progress in Artificial Intelligence (EPIA 05). Number LNAI 1036, (Springer-Verlag)
13. García, C., Hervás, R., Gervás, P.: Una Arquitectura Software para el Desarrollo de Aplicaciones de Generación de Lenguaje Natural. Procesamiento de Lenguaje Natural **33** (2004) 111–118
14. Lin, D.: Dependency-based evaluation of MINIPAR. In: Proc. of Workshop on the Evaluation of Parsing Systems, Granada, Spain (May 1998)
15. Meteer, M.W.: The generation gap: the problem of expressibility in text planning. PhD thesis, Amherst, MA, USA (1990)
16. McRoy, S., Channarukul, S., Ali, S.: A Natural Language Generation Component for Dialog Systems. In Cox, M., ed.: Working Notes of the AAAI Workshop on Mixed-Initiative Intelligence (AAAI99). (1999)
17. Schank, R.: Conceptual Information Processing. Elsevier Science Inc., New York, NY, USA (1975)
18. Lenz, M., Burkhard, H.: Case Retrieval Nets: Foundations, properties,implementation, and results. Technical report, Humboldt University, Berlin (1996)