Cross-Domain Analogy in Automated Text Generation

Raquel Hervás[1] and **Francisco C. Pereira**[2] and **Pablo Gervás**[1] and **Amílcar Cardoso**[2]

Abstract. Appropriate use of analogy in computer generated texts would constitute a great advantage in contexts where complex information needs to be communicated. This paper presents research aimed at improving the stylistic quality of the texts generated by a natural language generation system with the use of analogy. A generic architecture and a specific implementation for solving this problem are described. This implementation takes the form of a multiagent architecture in which several independent modules are connected, each one dealing with a subtask of the process - enriching domain information, finding structure alignment between domains, inserting analogies in a text generation pipeline. Initial results are presented, and several issues arising from the observed behaviour are discussed, with special attention to possible refinements of the proposal.

1 Introduction

Analogy is used frequently by humans when comunicating between them. Provided that our listeners have adequate knowledge of the target domain, using an apt analogy can be a more economical way of communicating information than actually explaining a whole set of facts about a given concept. If computers were to be capable of using analogies, this would constitute a great advantage in contexts such as pedagogical applications - where complex issues may need to be explained to a user -, or simply as an additional tool for communicating complex information in any kind of interactive setting. The task of identifying apt analogies is difficult even for human beings, and it is often considered to have a complex ingredient of creativity. However, the actual process of introducing an analogy into a given text is well within the possibilities of current natural language generation technology.

PRINCE (*Prototipo Reutilizable Inteligente para Narración de Cuentos con Emociones*) is a natural language generation application designed to build texts for simple fairy tales. The goal of PRINCE is to be able to tell a story received as input in a way that is as close as possible to the expressive way in which human storytellers present stories. To achieve this, PRINCE operates on the conceptual representation of the story, determining what is to be told, how it is organised, how it is phrased, and which emotions correspond to each sentence in the final output. Emotional content is added in the form of tags, to be realized as synthezised emotional voice [6]. PRINCE has been used as natural language generation front end of ProtoPropp [9], a system for automatic story generation.

[1] Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid, Spain, email: {raquelhb@fdi,pgervas@sip}.ucm.es

[2] Centro de Informática y Sistemas (CISUC), Universidade de Coimbra, Polo II, Pinhal de Marrocos, 3030 Coimbra, Portugal, email: {camara,amilcar}@dei.uc.pt

The research presented in this paper is aimed at improving the stylistic quality of the texts generated by the PRINCE system, by extending its capabilities to include the use of analogy. This is done by exploiting the potential of a lexical resource - such as WordNet - and structure mapping algorithms to enhance the output texts.

PRINCE is implemented using the cFROGS architecture [7], a framework-like library of architectural classes intended to facilitate the development of NLG applications. It is designed to provide the necessary infrastructure for the development, minimising the implementation effort by means of schemas and generic architectural structures commonly used in this kind of systems. cFROGS identifies three basic design decisions when designing the architecture of any NLG system: (1) what set of modules or tasks compound the system, (2) how control should flow between them, deciding the way they are connected and how the data are transferred from one to another, and (3) what data structures are used to communicate between the modules.

The flow of control information among the modules of PRINCE acts as a simple pipeline, with all the modules in a sequence in such a way that the output of each one is the input for the next. From a given plot plan provided as input to PRINCE, the text generator carries out the tasks of Content Determination, Discourse Planning, Referring Expression Generation, Lexicalization and Surface Realization, each one of them in an independent module.

Section 2 describes the lexical resources to be employed in this paper and previous work on analogy and structural alignment. Section 3 describes the basic architecture used for modelling the process of analogy generationg. The current multiagent implementation is presented in section 4. The experiments that have been carried out are described in section 5, and discussed in section 6. Section 7 outlines the conclusions.

2 Increasing lexical diversity

In PRINCE, the stage of lexical realization is done in a very simple way. Each concept in the tale has an unique associated tag, and for each appearance of a concept the corresponding word is used in the final text. This produces repetitive and poor texts from the point of view of the vocabulary.

To overcome this limitation, we have included a number of independent modules for expanding the number of lexical alternatives, thus allowing the enrichment of the realization process with lexical choice. Several strategies were envisaged and explored to provide those alternatives, from the more simple synonym substitution or the combination of this with the use of other kinds of semantic relations like hyponymy/ hypernymy, up to resorting to rethorical figures.

A requirement for these modules was the availability of adequate lexical sources. If synonym substitution could be carried on by means of a simple dictionary, more elaborated substitutions required richer

knowledge sources with adequate indexing mechanisms. WordNet [11] was found to be particularly well suited for this task as it is by far the richest and largest database among all resources that provide a mapping between concepts and words. Other relatively large and concept-based resources such as PENMAN ontology [1] usually include only hyponymy relations compared to the rich types of lexical relations presented in WordNet. For this reason, WordNet has been chosen as initial lexical resource for the development of the lexical choice task.

WordNet is an on-line lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. The most ambitious feature of WordNet is its attempt to organize lexical information in terms of word meanings, rather than word forms. English nouns, verbs and adjectives are organized into synonyms sets, each of them representing one underlying lexical concept. These synonyms sets - or *synsets* - are linked by semantic relations like synonymy or hyponymy.

Its organization by concepts rather than word forms allows WordNet to be used also like a knowledge source. The hyponymy/hypernymy relation can be considered equivalent to the "isa" one, and the gloss of the concepts contains extra information that in particular cases can be extracted automatically.

Using WordNet, lexical choice has been accomplished in a first phase by exploring synonymy and hyponymy relations. The method adopted is as follows. In the first appearance of a concept in a paragraph, the word from the system vocabulary is used. This is the word that has been chosen by the developer as the most descriptive one for the concept and explicitly written in the vocabulary. In the second appearance of a concept in a paragraph, its first hypernym is used. This hypernym is just a generalization of the concept, but the most specific one. In the rest of appearances of the concept, synonyms are used, one after the other, repeating them when the synonym list gets exhausted.

In a second phase, which is the focus of this paper, some rethorical mechanisms have been explored, namely Analogy, again with the help of WordNet.

2.1 Analogy, Metaphor and Structure Alignment

Metaphor and analogy are two cognitive mechanisms that have been recognized as underlying the reasoning across different domains[3]. Because of this, they play an indomitable role in creativity, thus calling our attention as a potential resource for the PRINCE project. Although no consensus has been reached in the current literature regarding a clear distinction between metaphor and analogy, it is clear that their mechanics share many commonalities. It is widely accepted in analogy research that many of the problems of metaphor interpretation can be handled using established analogical models, such as the structure alignment approach [8] [4]. The general idea behind this approach is that Metaphor (and Analogy) fundamentally result from an interaction between two domains (the vehicle and the tenor, in Metaphor literature). This interaction can be simplified as an isomorphic alignment (or mapping) between the concept graphs that represent the two domains. Thus, we see here a domain as being a semantic network (nodes are concepts; arcs are relations), and a mapping between two concepts (of two domains) results from the application of rules that rely on graph structure: if two nodes share the same connection to the same node, they form a potential mapping (triangulation rule [15]); if two nodes share the same connection to other two nodes that are forming a mapping, they form a potential mapping (squaring rule [15]). Since the domain mappings must be isomorphic (1-to-1), there may be many possibilities. n fact, the problem of identifying isomorphic graphs seems to fall in a crack between P and NP-complete complexity classes, if such a crack exists ([14], p. 181), and as a result, the problem is sometimes assigned to a special graph isomorphism complete complexity class. Given such perspective, the problem is clearly resource demanding, and getting the optimal solution seldom unrealistic. Therefore, our own approach follows a flood-fill probabilistic algorithm (see [13] for further details), such that no optimal result is guaranteed, but always takes acceptable time to retrieve a possibility.

For the PRINCE project, we are exploring the structure mappings with one particular realization template in mind: "X is the Y of Z" sentences [5].

X is the Y of Z

A mapping (say, from a concept X to a concept Y) produced by a structure alignment should emphasize some particular correspondence between two concepts, namely that, according to some perspective, the role that one concept has on one domain (say, the concept Y in the domain T) can be projected to its counterpart in the other domain (say, the concept X in Z). This is the rationale behind the "X is the Y of Z" expression, where Z is the domain in which X is integrated (from [5]). For example, "Freud is the father of Psychoanalysis" results from the the mappings `Freud` $\leftrightarrow$ `father` applied to the domains *Psychoanalysis* and *family structure*, respectively. One can find this template present in many more examples (e.g. "Brugges is the Venice of Belgium", "the Lion is the king of the jungle", "the eyes are the mirror of the soul", etc.). Our goal is therefore to apply this template (using a structure alignment algorithm) in order to get potentially creative text realizations. Thus, we always need two domain concept maps, one for the context at hand (i.e. partially describing the story that is being generated), another for the *vehicle* domain (the one from which to draw the *analogical* perspective). This in itself raises challenges (which domains to use? when? how to select a good mapping?) for which we have some ideas that we will sumarize in the discussion section.

As an initial motivating example, we tested with a single vehicle: the Greek deities domain, extracted from WordNet. It was obtained by isolating the subgraph representing the Greek deity taxonomy, enriched with a simple (algorithmical) extraction of relations from their glosses (to get knowledge such as "Aphrodite is the goddess of beauty", "Zeus is father of Aphrodite", "Aeolus is the god of wind", etc.). Whenever needed, our algorithm is able to map a part of the story under construction to this vehicle domain (thus providing expressions like "The princess was the Aphrodite of Royalty" or referencing king as the "The Zeus of Royalty"). This proved to be a very powerful mechanism, but it also presented strong limitations due to the fact that it contitutes a knowledge greedy algorithm.

3 A High Level Architecture for Generating Analogy in Text

The task of generating texts where analogies are used graciously involves a number of challenges that need to be tackled in separate modules. Let us assume that the task is addressed from the point of

[3] This claim is nowadays widely agreed, as metaphor is seen as a cognitive rather than a linguistic device. For an extensive *figurative versus literalist* analysis, we redirect the reader to [15]

[4] As semminal works in this area, we can name SME [4] and Sapper [15]

view of enriching a given text - represented in some kind of conceptual form that a generator is capable of rendering as text - with at least one analogy between a concept occuring in the text and some concept in another domain.

On one hand, there is the basic task of identifying the additional domain, the structural mapping that licenses the analogy, and the corresponding concept in the other domain. On the other hand there is the task of inserting the appropriate linguistic structure for the analogy in the original text, including both the task of building its linguistic rendering and selecting the actual location in the original text in which it is to be inserted.

3.1 Identifying the Target Domain and the Mapping

The task of identifying an appropriate additional domain as target domain for the analogy is quite complex. Given that the analogy is required to contribute to an act of communication, it is reasonable to say that in order to be appropriate as target domain in an analogy, a domain must be sufficiently well known to the intended readers of the text so as to require no additional explanation. This narrows down the set of possible domains. It also makes the solution to the problem depend on the particular reader for which the text is intended. Since this requires some means of representing the intended reader as part of the process of generation, for the time being, we consider the target domain as given. Further work must focus on exploring the role of reader representation on the choice of target domains.

Having established a particular target domain, the next challenge is to identify the relevant mapping. This involves an elementary operation of comparing two given domains to identify valuable mappings. This is a process of searching for structural analogies between two domains. In the context of looking for analogies for a given text, this requires additional preprocessing.

The conceptual representation provided for generating a text is not usually rich enough to constitute a domain with significant structure. Whereas a full representation of the complete domain that participates in a text may correspond to a complex taxonomical tree, the sort of input that a generator receives usually corresponds more to a set of facts - which may have been the leaves of the taxonomical tree. From this conceptual representation, the *context* in which the analogy is going to be set can be extracted. Let us assume that the context corresponds to some subset of the set of facts received as input to the generator. Before this can be checked for mappings against a full domain, it must be enriched to ensure that at least a significant part of its taxonomical structure is explicitly represented. This is achieved by expanding the context with the use of an *ontology*. Once this has been carried out, it can be compared with a different domain - the *target domain* - in search of structural analogies. The process should render a partial mapping between the enriched context and the target domain, where certain concepts in the enriched context are associated with certain concepts in the target domain. Each of the associations in this partial mapping is a candidate for establishing the analogy.

The basic architecture for this process is described in Fig. 1.

Some heuristic must be provided to select acceptable candidates for the analogy out of the complete set of (partial) mappings. One possible way to do this is to check how many actual relations determine the association - relations mirrored in both domains which provide the basis for the structural analogy. We are assuming that concept associations based on a higher number of relations are better candidates for analogy. Another help for this task could involve some kind of preliminary filter (e.g. if there are no mappable relations, then no mapping will be found) that would reduce the number of candidate target domains.

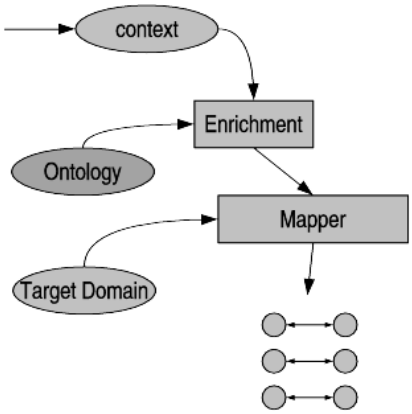

Figure 1. Finding partial mappings

3.2 Realizing the Analogy in the Text

The second task can be carried out in at least two different ways. One way is to respect the original text in its given form, and simply to build an additional sentence for conveying the analogy and inserting it at a chosen location. A more complex and richer solution is to add the corresponding message - represented in the same conceptual notation used for the original content of the text - to the input provided for the generator, and to let the generator convert the whole to a coherent text. This solution has the advantage of allowing the generator to reformulate the text surrounding the analogy so as to make the final resulting text linguistically and stylistically coherent.

For instance, take a simple text such as:

A princess lived in a castle. The princess loved a knight. She was the daughter of a king.

The first option described would correspond to inserting a sentence describing the analogy somewhere in the text:

A princess lived in a castle. The princess was the Aphrodite of royalty. The princess loved a knight. The princess was the daughter of a king.

For very simple generation processes the second option would present no difference. However, if the text generator has the capability of substituting certain repeated occurrences of the same concepts - as PRINCE does -, the text obtained for a conceptual representation of the first example might become something like:

A princess lived in a castle. She loved a knight. The princess was the daughter of a king.

In such a case, simple insertion of an additional sentence without touching the context would result in something like:

A princess lived in a castle. The princess was the Aphrodite of royalty. She loved a knight. The princess was the daughter of a king.

whereas adding the analogy at the conceptual level and then generating might result in:

> *A princess lived in a castle. She was the Aphrodite of royalty. The princess loved a knight. She was the daughter of a king.*

For our current purposes, we have opted for inserting a representation of the analogy at a conceptual level, and then generating a text for the resulting content.

When there is not enough information available, it is impossible to find the Z of the "X is the Y of Z". In that cases the relations that have produced the mapping can be used. If in the example shown there is no information about "the princess" belonging to "royalty", we can use the relation of "being daughter of" to produce an analogy like "The princess was the Aphrodite of the king".

4 A MultiAgent Implementation

The design of PRINCE is widely modular. In the same way, the roles that different parts take correspond to different processes and techniques. In order to allow an open perspective to the future as well as integrate multiple modules coming from different contributors, it became clear that a multi-agent platform would properly suit our needs. In this way, we can distribute different roles for different agents, each other being responsible for a special purpose task, acting autonomously and interacting only through clearly defined communication channels. This description coincides fairly with the Open Agent Architecture [2]. Such architecture is sufficiently open and modular to allow us implement and test the work presented in this paper as well as to make it easy to plug-in further functionalities. More precisely, we have a TextGenerator Agent, a WordNet Agent (for handling those queries to the database), a candidate reference agent (which gives sets of candidate references for a concept to whoever asks for them), a proxy agent - the OAA Facilitator agent that deals with requests/communications between different agents -, and two Analogy related agents. This agent society is shown in Fig. 2

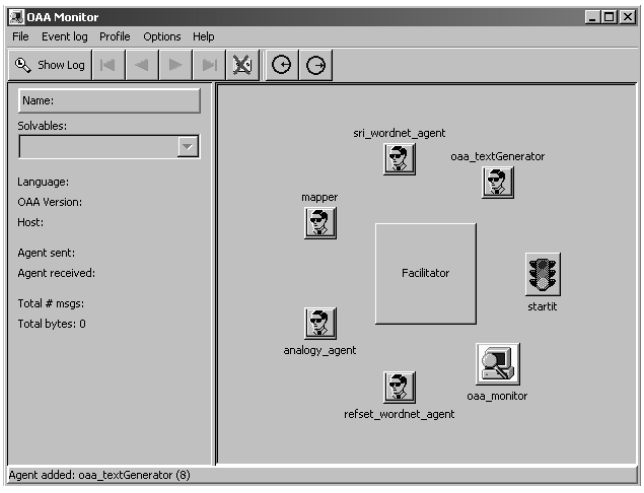

Figure 2. The agent society as seen in the OAA monitor

4.1 WordNet Agent

This agent was implemented in Prolog by Chris Culy [3] and is essentially a server for all the queries associated with WordNet database. For example, to ask for the gloss (the informal description of a concept) of a synset, one can send the following message to this agent:

$$wn\_g(SynsetID, Gloss).$$

where $SynsetID$ is the number identifying the synset, while $Gloss$ will have the reply that this agent will provide.

4.2 RefSet Agent

We developed this agent (also in Prolog) to provide the lexicalization module a set of candidate alternative lexicalizations. It receives the lexical item from the vocabulary (let us call it VBWord) for a concept, as well as its grammatical category (for example, for the concept *"dragon"*, VBWord would be the noun *dragon*). The VBWord will be the seed for searching for the corresponding WordNet synset.

WordNet is not organized according to individual *words*, it is organized according to *concepts*. Due to linguistic phenomena such as polysemy and synonymy, there is potentially a many-to-many mapping in which relates to concepts and words. This raises the important problem of Word Sense Disambiguation [10] (WSD), which has by itself deserved the attention of many researchers. At this point, WordNet provides some help: the *tag count* field for synsets. This field allows us to order, within a synset, which of the nouns is more *usual* in a generic corpus (in this case, the Brown Corpus[12]). Although this may not be the best corpus for our present purpose, the results showed it as a good choice for our first approach to this problem, since we avoided the much larger complexity involved in other WSD methods. In further explorations to this issue, we expect to build our own statistics out of story tale corpora.

After selecting the synset to explore, the RefSet agent gathers the set of synonym words (ordered by their individual tag counts) and the set of its hypernyms (ordered by their hierarchical distance - first *father*, then *grandfather*, then *grand-grandfather*, etc.). It then creates what we call a *context*: the set of relations from WordNet that are connected to the synonyms and hypernyms detected. In order to find a third list (the analogy list), it sends the words, as well as the context, to the analogy agent. This in turn delivers back (to the RefSet) the analogy proposals. The RefSet agent thus sends the set of those sets to whoever has called it (in this case, the lexicalization module). The message that RefSet agent expects to receive has the following structure:

$$refset(\quad Cat, Word, Context, \\ [RefSetSyns, RefSetHyps, RefSetAnalogies])$$

Currently, the RefSet agent uses services from the WordNet, the Facilitator, and the Analogy agents.

4.3 Analogy and Mapper Agents

The analogy agent is a proxy for the structure alignment agent (Mapper). It can optionally *enrich* the concept map of the VBWord (e.g. looking for relations in WordNet that are not considered in the knowledge base used by PRINCE), it loads the vehicle concept map for the Mapper agent. The Mapper agent looks for a structure mapping between the two domains and returns it to the Analogy agent, which fills the slots according to the "X is the Y of Z" template.

4.4 TextGenerator Agent

The TextGenerator agent is the one that deals with the NLG generation process, and can be considered as a wrapper for the original PRINCE module. This is the agent that sets off the flow of control information of the whole process. In a first step it initializes the mapper agent with the whole context of the tale that is rendering into text, producing the mapping between the domains involved. After that, the TextGenerator agent follows the usual pipeline control flow of PRINCE, interacting with the RefSet agent when needing the information of the different concepts in the tale.

5 Experiments

In order to test the analogical capabilities of PRINCE we have resorted to the use of domain data generated in the past for previous research on Metaphor and Analogy [15]. These data have two distinct advantages. On one hand they constitute a set of coherent domain data already tested for the existence of structural analogies. On the other hand, they were generated independently of the current research effort so they are less likely to be biased towards obtaining interesting results with the proposed method.

Out of the complete data set used in Veale's thesis, two well known domains have been used to test the analogy capabilities of PRINCE: Star Wars and King Arthur saga. The former has been chosen to represent a very simple story to be rendered by our generation system, including the most typical relations of the characters and elements of the domain. The latter is the domain used by the Analogy agent to find analogies with the first one. A conversion to Mapper representation was necessary, and in that process some amount of knowledge (from Tony Veale's Sapper) was left behind, namely relation weights, and some specific kinds of concepts (compound narrative relations, e.g. become_arthur_king, conceive_morgana_mordred). Thus, for the moment, we are focussing on the properties of caracters, objects and their first order relations within the story (e.g. have, friend_of, teach, loves, etc.).

The first step of the text generation in PRINCE is to obtain the possible analogies for the tale domain. The whole context of the story is sent to the RefSet agent so it can find out the analogies between the two domains used, and it is enriched by the Analogy agent using WordNet. For our simple Star Wars story part of the context is the following:

```
attr(obi_wan_kenobi, good)
have(luke_skywalker, light_saber)
teach(obi_wan_kenobi, luke_skywalker)
friend_of(luke_skywalker, han_solo)
loves(han_solo, princess_leia)
member_of(luke_skywalker, jedi_knights)
member_of(obi_wan_kenobi, jedi_knights)
gender(luke_skywalker,male)
gender(princess_leia,female)
...
```

The enriched graph of relations obtained from the initial context is mapped against the King Arthur saga relations. An extract of the domain information is the following:

```
isa(excalibur, weapon).
attr(excalibur, powerful).
have(king_arthur, excalibur).
gender(king_arthur, male).
gender(guinnevere, female).
friend_of(king_arthur, lancelot).
gender(lancelot, male).
loves(lancelot, guinnevere).
gender(merlin, male).
attr(merlin, good).
teach(merlin, king_arthur).
...
```

Some of the associations returned as part of a mapping are solely based on very simple general relations such as gender or *isa*. Such analogies are considered to be uninteresting and they are discarded by the generator. In this example the obtained mapping is shown in Table 1. For each association the list of relations that have produced the mapping is shown.

Cross domain association	Supporting Relations
$good \leftrightarrow good$	[attr]
$obi\_wan\_kenobi \leftrightarrow merlin$	[attr,teach,gender]
$luke\_skywalker \leftrightarrow king\_arthur$	[have,friend_of,teach,gender]
$light\_saber \leftrightarrow excalibur$	[attr,have]
$powerful \leftrightarrow hand\_held$	[attr]
$han\_solo \leftrightarrow lancelot$	[loves,friend_of,gender]
$princess\_leia \leftrightarrow guinnevere$	[loves,gender]

Table 1. Resulting mapping between StarWars and King Arthur domains

When using a concept during the generation process, PRINCE has the possibility of asking the RefSet agent for the analogy information of this concept.

For the time being, the complete set of analogies found is used, and each one is inserted in the original text after the first appearance of the corresponding concept. An extract of the resulting text in the Star Wars domain, using the analogies achieved, is the following:

> *Luke Skywalker was the King Arthur of the Jedi Knights. He had a light saber. The light saber was powerful. The light saber was the Excalibur of Luke Skywalker. Luke Skywalker was friend of Han Solo. Luke Skywalker was member of the Jedi Knights.*
>
> *Han Solo loved Princess Leia. He was the Lancelot of Luke Skywalker. She was the Guinnevere of Han Solo.*
>
> *Obi Wan Kenobi taught Luke Skywalker. Obi Wan Kenobi was the Merlin of the Jedi Knights. He was member of the Jedi Knights. He was good.*

6 Discussion

With respect to the classic pipeline structure of a simple natural language generator the introduction of a conceptual analogy would take place at the Content Determination stage. The process described above for identifying the target domain and the mapping would be carried out here. However, the alternative of introducing the analogy at the conceptual level opens some interesting possibilities. An analogy is based on a particular subset of relations that hold between elements in the context. If one is to say about someone "This man is the David Beckham of Java programmers", it must be true that this man is a Java programmer and that he is conspicuous for his success in comparison with other programmers. But in general, if one is to use the analogy, the actual relations that it is based on are usually left implicit. To spell them out weakens the stylistic effect of the analogy. This is one of the reason why the target domain must be presumed to be well known to the intended recipients. From the point of view of text generation, this implies that the introduction of the analogy may involve the suppresion from the final text of the explicit mention of at least some of the most obvious of the relations that support

it. The decision of how many of the supporting relations to suppress must also be based on some kind of model of the knowledge that the reader has. The minimum amount of relations required to make the analogy understandable must be retained. This issue is postponed until adequate means of modelling the reader can be introduced.

The problem of deciding where to place the linguistic realization of the analogy with respect to the rest of the text must be addressed at the stage of Discourse Planning. However, there are certain peculiarities involved in analogy that may require a certain degree of interaction between these two stages. An important question to take into account is that, if any of the supporting relations have been considered necessary for understanding the analogy, the linguistic realization of the analogy should not be placed in the text before those relations have been explicitly mentioned. An alternative approach would be to present the analogy first and then provide as an explanation the relations that actually support it.

In the description of the architecture presented in section 3.1, we considered a high-level approach to the identification and introduction of an analogy between a single concept from a given domain and a concept in a target domain. In the examples presented in section 5, it became apparent that, given two domains which are structurally analogous, the process described returns a number of associations between concepts in both domain. From a stylistic point of view, it is clear that a text in which all possible associations between the domains are included is overloaded with analogy. This suggests that the heuristics being considered for the selection and location of analogies must be revised in search for more natural results. Several issues must be addressed here.

To start with, it may be necessary to provide more strict criteria for filtering the set of associations returned by the analogy agent. The heuristics currently in use for this purpose were designed to select only one analogy for insertion in the text, and they are oriented to ensuring that no analogy is introduced unless it is supported by a minimal number of relations mirrored between the two domains. When several associations are possible, this restriction is not enough to reduce significantly the number of candidates.

Another important aspect is that the set of all analogies found as result of a mapping between two structurally analogous domains is itself so rich to constitute almost a parallel text in itself. In such cases, instead of inserting individual sentences describing each of the possible associations, it may be worthwhile to introduce a full sub-text describing the view of the original domain that corresponds to the target domain. Or to insert the analogy-related messages as groups of associations rather than as individual messages. This would correspond to considering the set of associations returned in a mapping as and addition to the conceptual content to be converted into text, susceptible of undergoing - as described above - stages of content determination - where some of the less interesting associations may be discarded - and discourse planning - where the remaining associations are regrouped, possible taking into account the target domain relations that may bind together concepts that appear in different associations. For instance, in the example presented above, it is clear that the messages describing King Arthur and Excalibur as analogous to Luke Skywalker and his light saber might be better presented if they are grouped together:

> *Luke Skywalker had a light saber. The light saber was powerful. He was the King Arthur of the Jedi Knights and the light saber was his Excalibur. ...*

However, this sort of arrangement may not be so good if it results in two completely different parallel texts. Some sort of intermediate clustering should take place during discourse planning, where elements bound together by some relation - like Luke and his light saber and/or Arthur and Excalibur - are grouped prior to establishing the analogy. This has happened by coincidence in the example text for Han Solo/Lancelot and Princess Leia/Guinnevere - and also for Merlin/Obi Wan Kenobi- due to the fact that they share a reciprocal relation. The heuristic should be refined to ensure that such effects are the result of explicit decisions rather than chance.

7 Conclusions and Further Work

The extension of the PRINCE system to include use of analogy shows acceptable results for instances where analogies are sought for a single concept. The multiagent architecture has proved to be a good solution for interconnecting the various resources and techniques that are required to solve the problem.

Further work is necessary to explore the extension of the functionality to instances where analogical equivalents are identified for more than one concept. The classic pipeline architecture of simple natural language generators provides a promising organization for the successive processes that would be involved in this task.

REFERENCES

[1] J. A. Bateman, R. T. Kasper, J. D. Moore, and R. A. Whitney. A General Organization of Knowledge for Natural Language Processing: the PENMAN upper model, 1990.

[2] A. Cheyer and D. Martin, 'The Open Agent Architecture', *Journal of Autonomous Agents and Multi-Agent Systems*, **4**(1), 143–148, (2001).

[3] Chris Culy. WordNet Agent. http://www.ai.sri.com/oaa/ contributions/wordnet, 2002.

[4] B. Falkenhainer, K. D. Forbus, and D. Gentner, 'The structure mapping engine: Algorithm and examples', *Artificial Intelligence*, **41**, 1–63, (1989).

[5] G. Fauconnier and M. Turner, *The Way We Think*, Basic Books, 2002.

[6] V. Francisco, R. Hervás, and P. Gervás, 'Analisis y síntesis de expresión emocional en cuentos leídos en voz alta', *Procesamiento de Lenguaje Natural*, (35), (2005).

[7] C. García, R. Hervás, and P. Gervás, 'Una arquitectura software para el desarrollo de aplicaciones de generación de lenguaje natural', *Procesamiento de Lenguaje Natural*, **33**, 111–118, (2004).

[8] D. Gentner, 'Structure-mapping: A theoretical framework for analogy', *Cognitive Science*, **7**(2), (1983).

[9] P. Gervás, B. Díaz-Agudo, F. Peinado, and R. Hervás, 'Story plot generation based on CBR', *Journal of Knowledge-Based Systems*, **18**(4-5), 235–242, (2005).

[10] N. Ide and J. Veroni, 'Word Sense Disambiguation: The State of the Art', *Computational Linguistics*, (24), 1–40, (1998).

[11] G. A. Miller, 'Wordnet: a lexical database for English', *Commun. ACM*, **38**(11), 39–41, (1995).

[12] W. Nelson Francis and H. Kucera, *Computing Analysis of Present-day American English*, Brown University Press, Providence, RI, 1967.

[13] F.C. Pereira, *A Computational Model of Creativity*, Ph.D. dissertation, University of Coimbra, 2005.

[14] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Reading, MA: Addison-Wesley, 1990.

[15] T. Veale, *Metaphor, Memory and Meaning: Symbolic and Connectionist Issues in Metaphor Interpretation*, PhD Thesis, Dublin City University, 1995.