The LOLITA User-Definable Template Interface

Marco Costantino
Laboratory for Natural Language Engineering, Department of Computer Science, University of Durham, U.K.

The development of user-definable templates interfaces which allow the user to design new templates definitions in a user-friendly way is a new issue in the field of information extraction. The LOLITA user-definable templates interface allows the user to define new templates using sentences in natural language text with a few restrictions and formal elements. This approach is rather different from previous approaches to information extraction which require developers to code the template definitions directly in the system.

After describing LOLITA as a general purpose base NLP System and other approaches to user-definable template interfaces that could be taken, the paper describes the design and the implementation of the LOLITA user-definable template interface. The performance of the interface is evaluated comparing the results with those produced by pre-defined financial templates produced by the LOLITA System.

Keywords: Natural Language Engineering, Information Extraction, User-definable template interfaces.

1. Introduction

Most of information extraction systems have been designed and tested within government agencies and the scientific community and very few real applications have been commercially successful. The emphasis has been on the improvement of the performance of the systems in terms of precision and recall. However, little progress has been done in making the systems user-friendly.

One of the main criticisms that can be made to many of the existing information extraction systems is that the users can't configure the systems to produce results (templates) which differ from those already available in the system. The *templates* are usually coded within the system and the user cannot modify the existing templates or add new ones without having to intervene directly on the system's code. For scientific competitions such as the MUC conferences [10], [11], [12], [13] this may be acceptable, but for real applications such as a financial application this problem is very relevant. With the advent of the first systems focusing on information extraction from the Internet (e.g. [2]), the issue is becoming more relevant.

The lack of flexibility of the current information systems has been also identified in the TIPSTER phase II project document [16], in which it is hoped that future systems will allow the definition of custom templates by the end-user. The document also defines specific standard objects and classes for the development of standardized components within a customizable information extraction system. The TIPSTER phase II document defines three different classes of objects for a customizable information extraction system:

- **ExtractionNeed**. This class should contain the input definition of the user, consisting of a formal specification (e.g. the template and slot names) and a narrative description describing the slot fill rules (e.g. the MUC-5 slot fill rules). This should be then translated by the system obtaining the CustomisedExtractionSystem.
- **CustomizedExtractionSystem**. This class should contain the system-specific procedures for extracting the user-defined templates from the source texts. These procedures should be created employing specific operations available in *CustomisedExtractionSystem.*

- **TemplateObjectLibrary**. This class should contain the system-specific rules for general concepts which might be used in the user's definitions of the templates such as *person*, *company* etc.

Although the architecture proposed in the TIPSTER phase II document does not describe how the user-definable systems should be implemented, the document represents a first step towards the development of a customizable information extraction system.

The *Hasten* system, which successfully participated in the MUC-6 competition [17], is a first example of a partially-customizable information extraction system. The interface is based on *example-patterns* corresponding to relevant fragments of source texts which can be entered by the user and will be used for producing the templates. Although the interface presents the advantage of allowing the user's definition of the slots, few problems arise in the definition of a new template:

- the template definition is still coded in the system. The user is allowed to enter slot definitions for the templates already coded in the system, but the definition of new templates must be done by modifying the system's code.
- the user is required to enter a considerable amount of example patterns for the definition of each slot. However, the problem is mainly caused by the fact that the system is based on pattern-matching techniques which require a considerable amount of patterns. For example, the total number of egraphs (Hasten's patterns) needed to define the MUC-6 management template [13] was 132 [17].

This work presents the LOLITA user-definable template interface, which allows the end-user of the system to enter new template definitions using natural language sentences with few restrictions and formal elements

The work is organised as follows. In section 2 we briefly discuss the main features of the LOLITA System. In section 3 we discuss the different approaches which can be taken for designing a user-definable template interface, while in section 4 we present the design and implementation of the LOLITA user-definable template interface. Finally, in section 5 we evaluate the results of the takeover template produced by the user-definable template interface, comparing the results with those produced using a pre-defined LOLITA takeover template.

2. The LOLITA System

LOLITA (Large-scale Object-based Linguistic Interactor Translator and Analyser) has been designed as a *general purpose* natural language

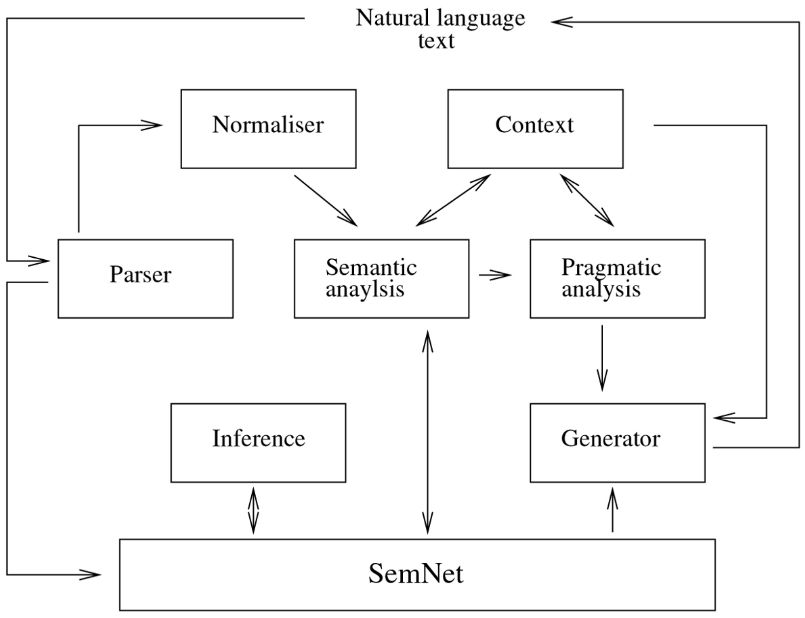

Fig. 1. The LOLITA system core.

processing system and has been under development at the University of Durham for the last nine years [14].

The approach taken for designing and implementing the system follows the lines of natural language engineering rather than those of computational linguistics. The NLE approach emphasizes the following aspects of engineering that should be considered when building a NL system: scale, feasibility, robustness, maintainability and usability [15].

The LOLITA system is written in the functional programming language Haskell (currently about 45,000 lines of code, corresponding to about 450,000 lines of code in an imperative language) and based on a large, WordNet-compatible semantic network, *SemNet*, (over 100,000 nodes), similar to a conceptual graph [20]. Its core, being the main part of the system around which individual applications are built, consists of 8 main modules (figure 1).

The semantic network consists of a hierarchy of nodes (concepts) connected with arcs. The nodes represent entities (*the company*) and events (*The company made losses*), while arcs represent relations (*A company IS A business*). Each node also has an associated set of control variables. There are about 50 different control variables. Some of the control variables are:

- **Rank.** This control gives the nodes quantification, i.e. individual, (*the loss Company XY made in the first quarter of '94*), universal (*every loss*), generic (*losses*, or *some lofsses*), existential, bounded existential etc.
- **Type.** This control value is very similar to grammatical qualification with few exceptions and additions: entity, relation, typeless, event, fact, greeting etc. The *relation* type mainly represents verbs, *attribute* represents adjectives and *entity* represents nouns.
- **Family.** This control groups nodes into the semantic "families", eg. living, animal, human, man-made, abstract, location, organization, human-organization etc. [14].

Concepts are linked with arcs such as **specialization_** (and its inverse, **generalization_**), or **instance_** (inverse **universal_**). Specialization links a set to a possible subset. For example, the concept of "*company*" is a specialization of the concept of "*business*" which is a specialization of the concept of "*enterprise*". The **specialization_** (**generalization**) link can be therefore used to specify hierarchies of concepts.

The **instance_** link allows to connect a concept to an instance of that concept. For example, the node corresponding to the organization "*ALPHA*" in the sentence "**ALPHA bought BETA**" will be connected with a **universal_** link to the set of all organizations, of which ALPHA is an instance.

These mechanisms allow the network to contain an elaborate "knowledge base" (i.e. encyclopedic "world" knowledge, linguistic knowledge) which can be expanded via the natural language interface that is part of the system.

Input natural language text is processed by various hierarchic modules and the result stored in the semantic network. The main processing phases are: *morphology*, *parsing*, *semantics* and *pragmatics* (figure 1).

- the **morphology** module is responsible for splitting the input text into words and smaller units and producing for each word a list of possible meanings of that word combined with their syntactic (noun, verb etc.) and semantic categories. The input is then passed to the parser;
- the **parser** determines the syntactic information contained in the source text. It performs a full grammatical analysis of the input text, recognising the role of each word in the sentence (e.g. subject, verb, adjective, object etc.). At this stage, the meaning of each of the words in the sentence can be still ambiguous and will be resolved by subsequent modules. Partial parsing is not currently implemented. If a sentence fails to be parsed, no result will be passed to the semantic analysis phase;
- the **semantic analysis** module associates the words with the appropriate meaning(s) and maps them onto the system's internal representation;
- finally, the **pragmatic analysis** module performs the disambiguation of the meaning of the words and type checking. Lexical ambiguities (e.g. different meanings of the same word) and anaphora are resolved using a series of preference heuristics, taking into account the *topic* which has been set for the

current text and the information in the *context*.

At this stage, the new knowledge can be stored in the semantic network and can be subsequently retrieved by the various applications.

To generate natural language output, the relevant part of the semantic network is fed to the generator component, which is capable of generating natural language output from the internal representation stored in the network [19]. The output from the generator can be varied according to a large set of parameters.

Various kinds of applications have been realised around the LOLITA core including: machine translation from Italian to English, English to Spanish, Language Tutoring [21], query application and contents scanning [14] and financial information extraction [7].

3. User-Definable Template Interfaces

The goal of a user-definable template interface is to allow the end-user of an information extraction system to add new templates to the system in a user-friendly way.

Source article:

FLORHAM PARK, N.J. (AP) – Generic drug maker Schein Pharmaceutical Inc. will acquire Marsam Pharmaceuticals Inc. for 240 million dollars, the two companies said.

The agreement calls for Schein to acquire all stock outstanding of Marsam at about 21 dollars a share. In May, Marsam, which makes injectable drug products, disclosed it had received unsolicited takeover offers in the range of 19 dollars a share. On Friday, Marsam shares closed at 19.3125 dollars, down 6.25 cents, in Nasdaq Stock Market trading.

```
Template: Takeover
     Company target: Marsam Pharmaceuticals Inc.
     Company predator: Schein Pharmaceutical Inc.
     Type of takeover: FRIENDLY
     Value: 240 million dollars
```

Fig. 2. The Takeover template.

A generic template such as the takeover template shown in figure 2 can be represented in the system with the following key elements:

1. the **template-name** which uniquely identifies the template among the others in the collection;
2. the **main-events** of the template, which represent the conditions under which the template has to be instantiated by the system;
3. the **slot-names** which uniquely identify each of the slots in the template;
4. the **slot-rules** which are used by the system to identify the relevant information for each of the slots.

The user-definable template interface will therefore need to allow the user to define these elements, the most difficult ones being the main-events and the slot-rules. Three different strategies can be taken for the definition of a user-definable template interface:

- A **menu-based environment.** In this case the user could construct the templates using pre-defined structures / components available in menus and using cut and paste techniques.
- A **example-based environment.** In this case the user would provide the system with a number of examples of relevant articles or of relevant articles' fragments. The system would then extract the relevant patterns which would be used in the extraction of the templates from the source articles.
- **Natural Language Text.** In this case the user is allowed to enter the full specifications for the main-event and each of the desired slots using sentences in natural language. The system will then translate this information into the appropriate template rules.
- **Interactive Natural Language Definition.** In this case the user would enter the template definitions using sentences in natural language. The system, however, would interact with the user to reduce the number of ambiguities in the template definitions. This could potentially lead to a dialogue-based definition environment.

Two main paths could be followed defining a menu-driven environment. A first possibility would be to provide very low-level primitives which could be employed by the user for the definition of the templates. However, very low-level primitives would make the environment rather complex and the user would need to spend a considerable amount of time for designing the templates. Another possibility would be to provide a high-level structure with specific objects already defined (e.g. company, person, etc.). The time needed for entering the template definition would be considerably lower. However, high-level structures would imply limitations to the expressive power of the users.

An example-based environment could potentially consist of two different situations. In the first case the user would provide examples of relevant articles for a specific template. The system would then automatically identify the appropriate information. The system developed by Collier [3, 4] is able to process a number of input texts and recognize the significant similarities between them, identifying a possible template for the extraction of the most important information.

In the second case, the user would provide some examples of relevant text fragments for the specific elements of the template, together with examples of filled templates (e.g. the Hasten MUC-6 System [17]). The text fragments and the templates entered by the users would be subsequently used by the system for analysing the source articles. However, in a pure example-based environment the user would be required to enter a considerable number of examples (either source articles or text fragments) which would drastically increase the time needed for the definition of the template elements. For example, the definition of the MUC-6 management scenario template using the Hasten system required 132 example patterns [17].

Given the limitations of the approaches described above, the LOLITA user-definable template interface has been designed to accept natural language input.

4. The LOLITA User-Definable Template Interface

The LOLITA user-definable template interface has been designed to process templates definitions in free natural language, using specific formal elements designed to reduce the ambiguity to the input sentences.

The definition of the interaction way between the user and the system, which corresponds to the definition of the class of objects "*ExtractionNeeds*" described in the TIPSTER phase II document, has been done by analysing the results of an experiment carried out by potential users of the system. The test required the potential users to describe a generic takeover template using sentences in natural language. More specifically, the users were asked to describe the *main condition* and the specific *slot rules*. These were the main aims of the experiment:

- to identify how easy it is for the user to define the templates using unconstrained input natural language text;
- to establish how easy it would be for the system to understand such unrestricted input definitions.

The ultimate target was to identify the optimum compromise between the two. The analysis of the results suggests that allowing complete freedom for the user can lead to a difficult situation for both the user and the system:

- the user can find it difficult to express the template definitions using unrestricted natural language text without the support of any formal element;
- the unrestricted natural language input can be rather difficult to process for the system and a relevant number of ambiguities can be found in the template definitions. These ambiguities mainly concern the resolution of *anaphora*. In other words, how to resolve the relations between objects and events in the template-condition and in the slots (coreference resolution).

Four main styles in the template definitions have been identified in the experiments carried out:

1. **The use of questions.** e.g. *What was the cost of the takeover?*
2. **The use of Noun-Phrases.** e.g. *The cost of the takeover.*
3. **The use of statements.** e.g. *A company acquires another company.*
4. **The use of variables.** e.g. *Company X acquires company Y.*

The use of *questions* in the definition has not been chosen for the user-definable interface. This is because the potential users were unable to use the question for defining all the elements of the templates, but they were able to express the same definitions using noun-phrases or statements in place of questions. Allowing the user of questions would have therefore meant introducing additional ambiguities which could have been avoided. *Noun-phrases* and *statements* have been chosen for entering the template definitions, depending on the template element to be defined.

The most important characteristic of the user-interface is that it allows the use of *variables* in the template definitions. This drastically reduces the amount of ambiguities in the definitions. The definition of the slot "*COMPANY_PREDATOR*" in the takeover template will therefore be:

```
Template Condition: V=COMPANY1
                    acquired V=COMPANY2
COMPANY_PREDATOR:   V=COMPANY1
```

The contents of the slot are clearly defined and no ambiguities arise from the definition. The same definition without the use of variables could have been, for example:

```
Template Condition: A company acquired
                    another company.
COMPANY_PREDATOR:   name of the company
                    that is purchasing
```

Differently from the previous definition, this second definition presents difficult points for both the system and the user. Firstly, the system would have to identify the specific company to which the user is referring. This is not necessary in the processing of the slot definition "*V=COMPANY1*", where the system can immediately identify the specific company, which corresponds to the variable. Secondly, the user may find it difficult to express the concept of "*company predator*" using a natural language sentence, while the definition of the slot using the variable "*V=COMPANY1*" is immediate.

Three different variables have been introduced. These variables, **formal elements**, have been designed to reduce the amount of possible ambiguities in the template definitions without reducing the user's expression power. The **formal elements** are:

- the **name of the template**, which distinguishes the template among the other templates in the system;
- the **template variables**, which identify the elements of the main-events, which will be later used in the definition of the slot-rules.
- the **slot-names**, which identify the specific template's slots and can be used in the definition of other slot rules to refer to the information contained in the previous slots.

The user can enter a new template definition using sentences in natural language, which follow a specific syntax which is here discussed in detail. Five elements must be entered for defining a user-defined template such as the takeover template definition shown in figure 3: the template name, the variables, the main-events, the slot-names and the slot-rules.

The name of the template must be defined using any sequence of capital letters or numbers according to the following rules:

- the name must start with the string "T=", for example "T=TAKEOVER";
- the name must be a single word. If more words are necessary, they must be joined with the character "–", for example: "T= MARKET-MOVEMENT". The name can be used in the definition of the slots to refer to the template as a whole.

Similarly, the variable name must be defined as follows:

- the name must be entered in capital letters and starting with the string "V=", for example: "V=COMPANY1", "V=VALUE", etc.
- the name must be a single word. If more words are needed, they must be joined with the character "-", for example: "V=COMPANY-ONE".

The user must define the variables (give a type) using *True-False* assertions, noun-phrases are not permitted. A valid definition of a variable is the definition of the variable "V=COMPANY1" in figure 3.

Once the variables have been defined, the user must enter the template main-event condition. This is used by the system to decide when the template has to be created. The main-template condition must be entered in the form of a *True-False assertion*. Noun phrases are not allowed unless they describe an event. Variables which have been previously defined can be used in the main-event and can be subsequently employed

```
Template-name:        T=TAKEOVER
Variables:            V=COMPANY1 is a company.
                      V=COMPANY2 is a company.
                      V=VALUE is money.
Template main-event:  V=COMPANY1 acquired V=COMPANY2.
                      V=COMPANY1 acquired V=COMPANY2 with V=VALUE.
                      The acquisition of V=COMPANY2 by V=COMPANY1.
                      The V=VALUE acquisition of V=COMPANY2 by V=COMPANY1.
                      V=COMPANY1 paid V=VALUE for V=COMPANY2.
                      V=COMPANY1 acquired a majority stake in V=COMPANY2.
                      V=COMPANY1 took full control of V=COMPANY2.
Definition of slots:
S=COMPANY-PREDATOR:   V=COMPANY1

S=COMPANY-TARGET:     V=COMPANY2

S=TYPE-OF-TAKEOVER:
  String-fill: HOSTILE   T=TAKEOVER is hostile.
  String-fill: FRIENDLY  otherwise

S=VALUE-OF-TAKEOVER:  The cost of T=TAKEOVER.
                      V=VALUE
S=BANK-ADVISER-PRED:  The adviser of V=COMPANY1.

S=BANK-ADVISER-TARG:  The adviser of V=COMPANY2.

S=EXPIRY-DATE:        The date of expiry of T=TAKEOVER.

S=ATTRIBUTION:        The person or the company that announced T=TAKEOVER.
                      The person or the company who said something about
                      T=TAKEOVER or said something about S=COMPANY-PREDATOR
                      or said something about S=COMPANY-TARGET or said
                      something about S=TYPE-OF-TAKEOVER or said something
                      about S=VALUE-OF-TAKEOVER or said something about
                      S=BANK-ADVISER-PRED or said something about
                      S=BANK-ADVISER-TARG or said something about EXPIRY-DATE.

S=CURRENT-STAKE-PRED: The stake that V=COMPANY1 owns of V=COMPANY2

S=DENIAL:             The person or company who denied T=TAKEOVER
                      or denied COMPANY-PREDATOR or denied the
                      COMPANY-TARGET or denied TYPE-OF-TAKEOVER or
                      denied S=BANK-ADVISER-PRED or denied
                      S=BANK-ADVISER-TARG or denied S=VALUE-OF-TAKEOVER
                      or denied EXPIRY-DATE.
```

Fig. 3. The takeover template as defined for the template user-interface.

in the slot-rule definitions to refer to specific information in the main-event. A legal main-event condition is shown in the takeover template in figure 3.

The next step is the definition of the slot-names, which must be defined according to the following rules:

- the name must be entered in capital letters and starting with the string "S=", for example: "S=FIRST-COMPANY", "S=ATTRIBUTION" etc.
- no spaces are allowed. If more words are needed, they must be joined with the character "-", for example: "S=BANK-ADVISER-PRED".

Each slot-name is associated with one or more slot-rules, which are used by the system to extract the relevant information from the source documents. The slot rules can be defined using a *noun-phrase* describing the information which has to be extracted for the specific slot. Slot-rules can make use of the name of the template for referring to the template as a whole. This is useful if the user wants to refer to the general concept of the template, for example "T=TAKEOVER", for the takeover template shown in figure 3. For example, the following slot from the takeover template shown in figure 3 refers to the name of the template:

```
S=VALUE-OF-TAKEOVER:
```

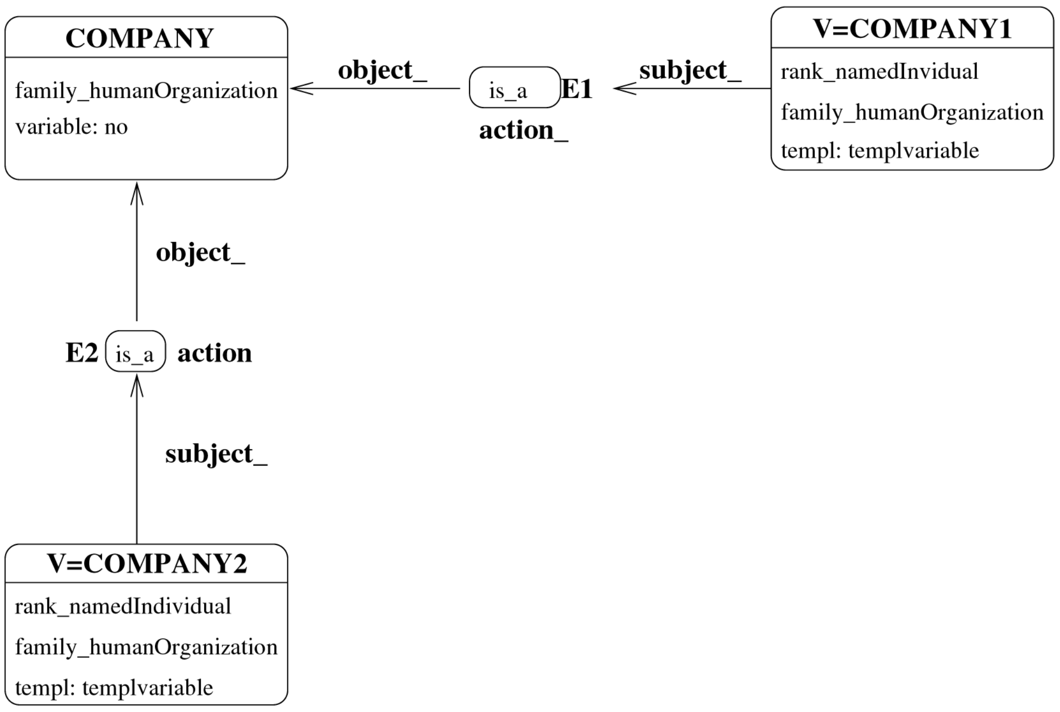

Fig. 4. The processing of a the variable "V=COMPANY1 is a company."

```
        The cost of T=TAKEOVER
```

The slot rules can also refer to the template variables, with the condition that the variable must have been used in the definition of the main-events, for example the following slot from the takeover template shown in figure 3:

```
S=BANK-ADVISER-PRED:
        The adviser of V=COMPANY1
```

The slot rules can also refer to the contents of slots which have already been defined by citing the slot name, for example the following slot-rule would be legal:

```
S=BANK-ADVISER-TARG:
        The adviser of S=COMPANY-TARGET
```

4.1. Implementation of the User-Definable Interface in the LOLITA System

The way in which templates are filled by the user-definable interface is rather different from how templates defined in the LOLITA system (e.g. the LOLITA financial templates [7]) are processed.

The most important difference is that no code describing the templates rules is available in the system. The user-defined templates are filled by the system using the *inference system* which matches the templates definitions against the knowledge contained in the semantic network and, in particular, the new knowledge acquired with the analysis of a source article. Therefore, the *inference system* identifies entities and events which satisfy the template rules stored in the semantic network corresponding to the *variables*, the *main-conditions* and the *slot-rules* definitions.

The first step taken by the user-definable interface is to process the template definitions supplied by the user ("*ExtractionNeeds*"). This corresponds to the operation "*Customise (ExtractionNeed)*" of the TIPSTER phase II document [16]. The *template-name*, the *variables* (figure 4), the *main-conditions* (figure 5) and the *slot rules definitions* are processed and stored in the semantic network.

The inference system will then try to match these questions against the new information acquired from the processing of a source article. For example, for the main-event shown in figure 5:

```
V=COMPANY1 acquired V=COMPANY2
```

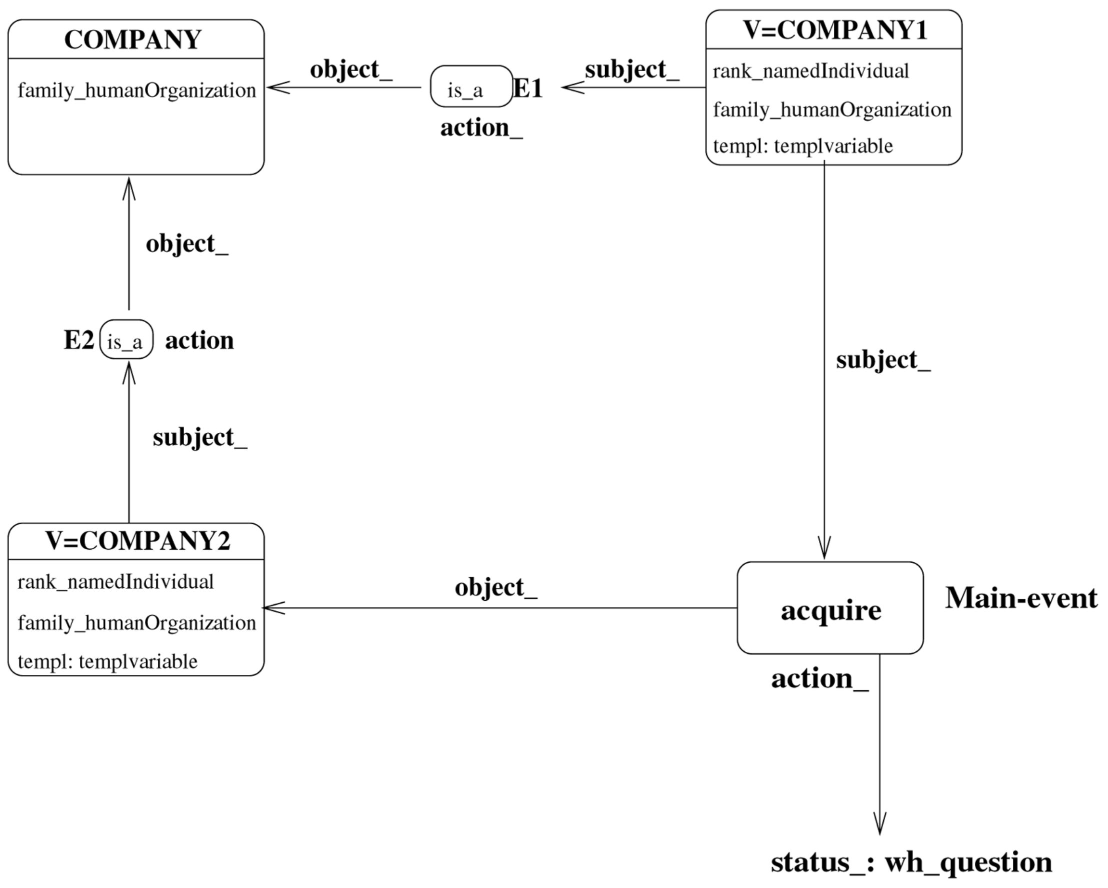

Fig. 5. The processing of the main-event “V=COMPANY1 acquired V=COMPANY2.

the inference system will recognize that an event such as:

```
Fiat purchased Renault
```

is a relevant one, because of the fact that the action is compatible with “acquire” and the subject and object can be matched against the variables “V=COMPANY1” and “V=COMPANY2”.

Figure 6 shows the representation of the main-event and the candidate event “Fiat bought Renault”. The inference system tries to match each of the components of the candidate event onto the main-event.

The inference system will therefore look for an event which satisfies the following condition:

$\exists$ *V=COMPANY1, V=COMPANY2. Acquire(V=COMPANY1,V=COMPANY2)*

Once the candidate events have been identified, these can be used by the inference system for searching for concepts which match the slot-rules.

Inference and the Variables

The variables are filled in by the inference system as part of the processing of the main-events. Therefore, specific calls to the inference system for locating information which corresponds to the variables are not necessary.

Inference and the Slots

The slot-rules definitions entered by the user can be subdivided into two different categories:

- rules which refer only to a specific variable used in the main-event, for example:

  ```
  S=VALUE-OF-TAKEOVER:   V=VALUE
  ```

  This kind of slots is filled with the concepts which have already been identified for the specific variable.

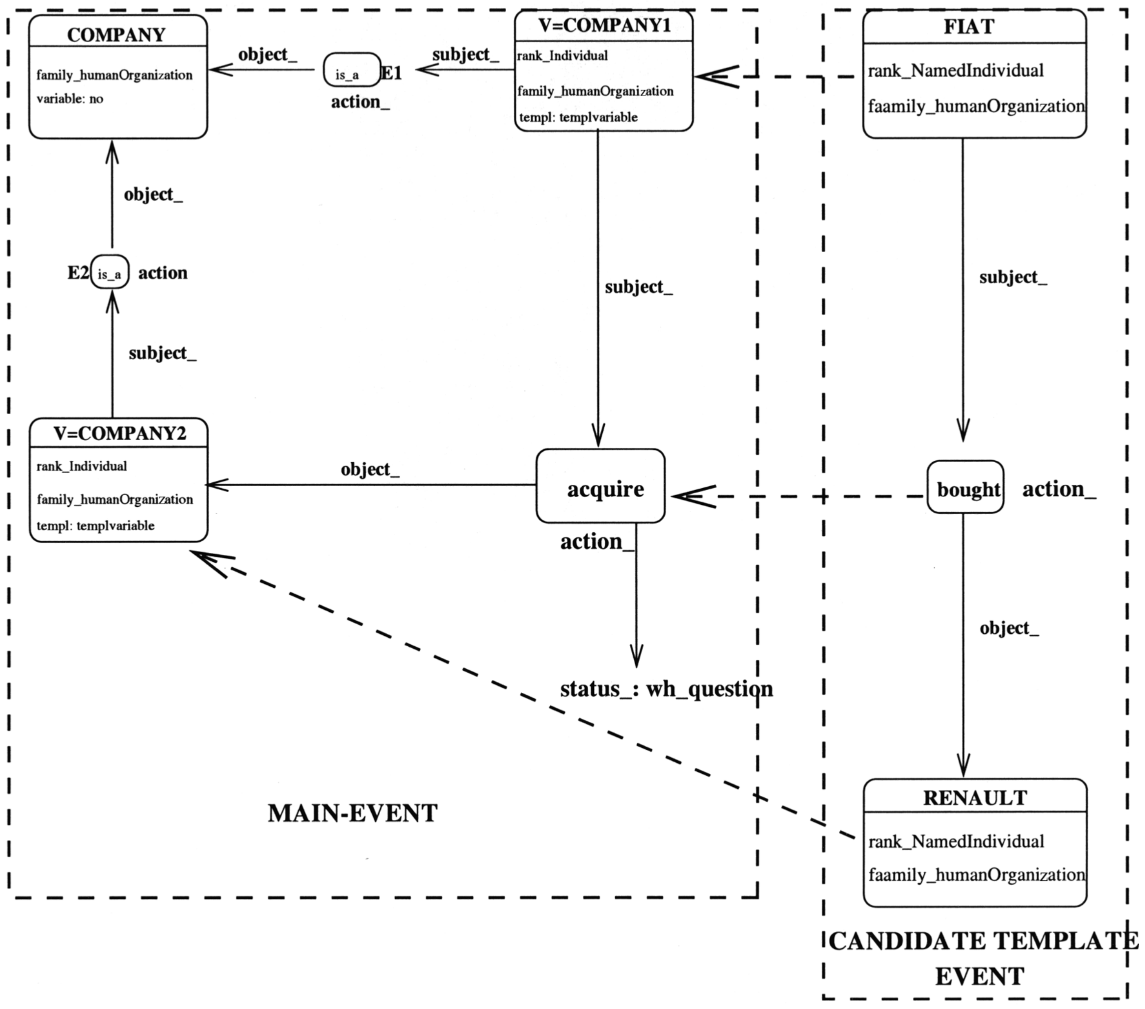

Fig. 6. Identification of candidate main-events by the inference system.

- rules which refer to specific variables, the template-name or other slot-names but adding additional conditions, for example:

```
S=VALUE-OF-TAKEOVER:
     the cost of the T=TAKEOVER
```

  In this case, the inference system will be called again and will look for any event or entity which matches the slot-rules.

Figure 7 shows the takeover template extracted from a source financial article. The template has been produced using the takeover template definition shown in figure 3.

Reuters Holdings yesterday announced that it acquired Teknekron Software Systems for 125.1 million dollars cash. Teknekron, a software supplier and systems integrator based in Palo Alto, California with a workforce of 200, had turnover last year of 38.7 million dollars and pre-tax profits of 8.2 million dollars. Net assets at the end of 1992 were 3.6 million dollars. Reuters has 212,000 information outlets worldwide, including 350 of the latest digital Triarch systems. Under the deal, which has to clear both the US and UK regulatory authorities, Teknekron will retain operational control of the company. Two non-executive directors from Reuters will join the Teknekron board. Teknekron's management will also benefit from a stock appreciation plan, similar to a share option scheme.

Template produced by the LOLITA system:

```
<T=TAKEOVER> :=
  S=TYPE-OF-TAKEOVER: FRIENDLY
  S=VALUE-OF-TAKEOVER: "Million 125.1 dollar cash."
  S=ATTRIBUTION: "Reuters Holdings."
  S=COMPANY-TARGET: "Teknekron Software Systems."
  S=COMPANY-PREDATOR: "Reuters Holdings."
```

Fig. 7. An example of the takeover template produced by the user-definable template interface using the takeover template definition shown in figure 3.

5. The Performance of the User-Definable Template Interface

The evaluation of the user-definable template interface is based on an experiment similar to the one carried out for the design of the interface (see section 4). A total of 14 potential users of the system were asked to describe a takeover template using the specific syntax of the user-definable interface (see section 4). Two were the main aims of the experiment:

- to evaluate how difficult is for the users to define a template using the user-definable template interface;
- to evaluate the performance of the user-definable templates on a evaluation set articles.

In section 5.1 we discuss the evaluation procedure and the results of the first evaluation aim, while in section 5.2 we discuss the results of the second evaluation aim.

5.1. Evaluation of the Design and Usability of the Interface

A heterogeneous group of 14 potential users of the system were asked to submit the definition of a takeover template following the rules of the user-definable template interface. The templates could be submitted using a specific WWW server which included the full description of the task and the full instructions of how to enter new templates definitions using the user-definable template interface[1]. The users were also required to enter the total time they spent reading the instructions regarding the user-definable interface and the time spent entering the definition. Figure 8 shows an example of the user-defined takeover template submitted by one of the 14 users. Specific measures have been introduced for the evaluation of the 14 templates definitions.

```
Template_Name:      T=TAKEOVER
Variable_1:         V=COMPANY1 is a company
Variable_2:         V=COMPANY2 is a company
Variable_3:         V=VALUE is money
Main_Event_1:       V=COMPANY1 bought V=COMPANY2
                    with V=VALUE
Main_Event_2:       V=COMPANY1 bought V=COMPANY2.
Slot_Name_1:        S=COMPANY-PREDATOR
Slot_Rule_1.1:      V=COMPANY1
Slot_Name_2:        S=COMPANY-TARGET
Slot_Rule_2.1:      V=COMPANY2
Slot_Name_3:        S=VALUE-OF-TAKEOVER
Slot_Rule_3.1:      V=VALUE
Slot_Rule_3.2:      The cost of T=TAKEOVER
Slot_Name_4:        S=ATTRIBUTION
Slot_Rule_4.1:      The company that announced
                    T=TAKEOVER
Time_Instructions:  15 minutes
Time_Form:          10 minutes
```

Fig. 8. An example submission of a user-definable template.

5.1.1. The SlotError Measure

The first measure, called **SlotError**, measures the difficulty in entering a user-definable template and depends on the number of errors in the forms submitted by the users. The higher the number of these errors, the more difficult it is for the user to define a template using the user-definable template interface.

An error occurs when the user defines an element of the template which cannot be correctly interpreted by the system and leads to a missing or incorrect template or slot. We therefore compute the total number of slots wrongly defined by the users and we relate it to the total number of slots defined. In this way we obtain a measure of the number of slots containing errors, which will never be filled by the system or will be filled incorrectly, of the total number of slots defined by the users. This measure, called *SlotErrors* is defined as follows:

$$SlotErrors = \frac{total\ number\ of\ slots\ containing\ errors}{total\ number\ of\ slots\ defined} \times 100.$$

The 14 templates defined by the users comprised a total of 70 slots. A total of 4 slots containing errors were found in the templates entered by the 14 users. We compute the measure *SlotErrors* as follows:

$$SlotError = \frac{4}{70} \times 100 = 5.71\%.$$

The **SlotError** measures shows that the 14 users incorrectly defined the 5.71% of the key information in the templates. We therefore conclude that the user-definable interface is rather easy to

[1] The WWW server can be found at: `http://www.dur.ac.uk/~dcs3mc/udeval`

use, since the 14 users were able to define a template with a very low rate of errors. This appears even more relevant considering that the majority of the users who entered the definitions of the takeover template were unfamiliar with natural language processing, information extraction and user-definable template interfaces.

5.1.2. The Average Time of Entering the Definitions

The second measure employed for the evaluation of the user-defined templates is the average time taken by the users in entering the user-defined takeover template.

Overall, the average user took 15.71 minutes for entering the definition of a takeover template. We consider this time particularly interesting, since it refers to users with no prior knowledge of natural language processing, information extraction and user-definable template interfaces.

Cowie Group, the car leasing and motor trading company, yesterday announced a big expansion of its bus operations with the 29.9 million pounds acquisition of Leaside Bus Company, the subsidiary of London Regional Transport (LRT). The deal, involving a 25.5 million pounds cash payment and 4.4 million pounds to settle intra-group loans, will enlarge Cowie's bus fleet from 128 vehicles to more than 600 and is expected to lead to a fourfold sales increase.

'We paid slightly more than we wanted to, but it was worth it for the enormous growth that it promises,' said Mr Gordon Hodgson, chief executive. The acquisition follows four months of talks between LRT and Cowie, which has been seeking a larger stake in the London bus network for more than two years.

At present, the group's bus and coach operations are dominated by Grey-Green-acquired 14 years ago – which serves 13 bus routes in London and employs 450 drivers. Leaside, by comparison, has a workforce of about 1,800 and operates 28 routes.

Mr Hodgson, who is meeting Leaside managers today, said he was determined to introduce private sector efficiency to the business, which last year made profits of just 607,000 pounds on turnover of 43 million pounds. In the same period, Grey-Green made profits of 1.6 million pounds on sales of 14.4 million pounds. Cowie shares fell 3 1/2 p to 218 1/2 p yesterday – a new low for the year.

Fig. 9. A relevant article of the evaluation set.

5.2. Evaluation of the Performance of the User-Defined Templates

The performance of the user-definable template interface has been evaluated scoring the results of the information extracted for the user-defined takeover shown in figure 3 by the system from an evaluation set of 55 financial articles (25 relevant takeover articles and 30 non-relevant financial articles). In figure 9 a relevant takeover article from the evaluation set is shown.

The scores have been computed using a modified version of the MUC-6 scoring program which was released to the developers of the MUC-6 systems [1]. The scoring program matched the templates produced by the system for each article against the corresponding key templates producing a summary reporting the *precision*, *recall* and the combined *'F' measure*[2].

Figure 10 shows the overall results for the 55 articles of the evaluation set. The final re-

[2] *Precision*, *recall* and *'F' measure* are standard measures employed in information retrieval and information extraction to evaluate the performance of a system. *Precision* can be thought of as the ratio of the number of relevant documents retrieved to the total number of documents retrieved [18]. The MUC *precision* measure was adapted for information extraction systems:

$$precision = \frac{correct\ +\ (partial\ \cdot\ 0.5)}{number\ of\ actual\ answers}.$$

Recall is the ratio between the number of relevant documents retrieved and the total number of relevant documents (both retrieved and not retrieved) [18]. The MUC *recall* measure was adapted for information extraction systems:

$$recall = \frac{correct\ +\ (partial\ \cdot\ 0.5)}{possible}.$$

Finally, the *'F' measure* represents a way to combine the precision and recall measures into a unique value and was first introduced by van Rijsbergen [18]. The *'F' measure*, as combination of precision and recall, gives a value that falls between them. The β parameter in the *'F' measure* represents the relative importance given to recall over precision and in the case recall and precision are of equal weight, β assumes value 1.0. The *'F' measure* presents a higher value if precision and recall are closer to the center of the recall-precision graph than if they are at the extremes of it. For example, if a system has precision and recall both of 50 per cent, the *'F' measure* will be higher than a system that has recall of 20 per cent and precision of 80 per cent. This is also because the aim of the formula is to direct developers towards an improvement of both recall and precision.

$$F\ measure = \frac{(\beta^2 + 1.0) \cdot P \cdot R}{(\beta^2 \cdot P) + R}.$$

sults show that the system's overall performance measures were:

```
                         P&R    2P&R   P&2R
F-MEASURES               37.44  46.17  31.4

OVERALL PRECISION:       55%
OVERALL RECALL:          28%
```

These measures have been compared with the results produced by the pre-defined financial takeover template over the same set of source financial articles. The pre-defined takeover template represents a normal LOLITA template. Differently from the user-definable template interface approach, the template definition is coded directly in the system and the template is filled searching the semantic network for the relevant information for each of the slots, rather than using the inference system. The LOLITA financial templates have been fully described in [5, 6, 7, 8, 9]. The performance of the pre-defined takeover template over the same set of articles is the following:

```
                            P&R    2P&R   P&2R
F-MEASURES                  51.03  57.41  45.93

OVERALL PRECISION:          63%
OVERALL RECALL:             43%
```

The performance of the pre-defined takeover template is higher than the equivalent user-defined takeover template. The overall loss of performance of the user-definable takeover template compared to the pre-defined takeover template is therefore:

$$Loss\ of\ performance = 100 - \frac{F - Measure\ user - definable\ templates}{F - Measure\ pre - defined\ templates} \times 100$$

$$Loss\ of\ performance = 100 - \frac{37.44}{51.03} \times 100 = 26.63\%$$

The user-defined takeover template presents a 27% loss of performance compared to the pre-defined template.

The loss of performance is mainly due to the difference in the way the user-defined templates are produced by the system. While the main-events and slot-rules definitions for the pre-defined takeover templates are directly coded in the system, the equivalent definitions for the user-definable template are instead obtained from the analysis of the source text of the template definition. This additional step relies directly on the analysis of the source text performed by the LOLITA core system. If a main-event or slot-rule definition is not correctly processed by the LOLITA core system, the user-definable interface will be unable to correctly identify the relevant information in the source article.

```
Report for the user-definable templates finalEvalUD2:

------------------------+-------------+--------------+-----------------------
SLOT            POS    ACT| COR PAR INC | MIS   SPU  NON| REC PRE UND OVG ERR SUB
------------------------+-------------+--------------+-----------------------
takeover         36     23|  14   0    2|  20     7    0|  39  61  56  30  67  13
  companytar     36     23|  11   0    5|  20     7    0|  31  48  56  30  74  31
  companypre     35     23|  11   0    5|  19     7    0|  31  48  54  30  74  31
  typetakeov     36     23|  14   0    2|  20     7    0|  39  61  56  30  67  13
  value          28      4|   4   0    0|  24     0    0|  14 100  86   0  86   0
  badviserpr      0      0|   0   0    0|   0     0    0|   0   0   0   0   0   0
  badviserta      0      0|   0   0    0|   0     0    0|   0   0   0   0   0   0
  expirydate      0      0|   0   0    0|   0     0    0|   0   0   0   0   0   0
  attrib          9      2|   1   0    1|   7     0    0|  11  50  78   0  89  50
  currentsta      0      0|   0   0    0|   0     0    0|   0   0   0   0   0   0
  denial          0      0|   0   0    0|   0     0    0|   0   0   0   0   0   0
------------------------+-------------+--------------+-----------------------
ALL OBJECTS     144     75|  41   0   13|  90    21    0|  28  55  63  28  75  24
------------------------+-------------+--------------+-----------------------
                                                       P&R       2P&R      P&2R
F-MEASURES                                           37.44      46.17     31.49
```

Fig. 10. The final score report for the user-defined takeover financial template. The report has been automatically generated by the MUC-6 scorer. The most important measure on the table is the first F-Measure from the left, while the other two are weighted towards precision and recall respectively.

Although the drop in performance (27%) can appear significant at the first sight, the time taken for the development of both the pre-defined and the user-definable templates must be taken into account. The pre-defined takeover template has been defined and coded within the LOLITA System in a period of time of about 8 months. This period of time included understanding how to code new template definitions in the LOLITA System, identifying the relevant rules for the takeover template, coding, compiling and testing the template definition.

Differently, the implementation of the user-defined takeover template, once the user-definable interface had been coded in the LOLITA System, required a significantly lower amount of time, which can be quantified in a total of about half a month.

We can therefore compare the two figures as follows:

$$\frac{time\ for\ defining\ the\ user-defined\ takeover\ template}{time\ for\ defining\ the\ pre-defined\ takeover\ template} = \frac{0.5\ months}{8\ months} \times 100 = 6.25\%$$

The above figures show that defining the takeover template using the user-definable template interface required a time at least 24 times lower than for defining the pre-defined template and led to a 27% loss of performance.

The figures show that user-definable templates are a feasible way of defining new templates within the LOLITA System, and require a time sensibly lower than for coding the new templates within the system. The absolute *precision* and *recall* figures can appear low. However, it is important to notice that they were obtained without any specific improvement of the LOLITA knowledge base for the financial domain. In addition, improvements are currently being carried out on the LOLITA core which should progressively improve its performance.

6. Conclusions

In this paper we presented the LOLITA user-definable template interface, which allows the user to easily define new template definitions using sentences in natural language. The evaluation of the interface has shown that inexperienced users found it very easy to enter new templates definitions in the amount of time limited to 15.71 minutes. The loss of performance of the user-defined templates compared to hand-coded templates is justified by the much lower time required for defining a template. The application can be useful for financial operators, who have to deal with the increasing quantity of qualitative information available today. By being able to quickly enter a new template definition and extract the relevant information from a large quantity of source articles, the operators can gain knowledge which can be extremely useful for taking appropriate financial decisions. This knowledge would otherwise be lost due to the financial operator's lack of time.

References

[1] N. Chincor and G. Dungca, "The Scoring Method for MUC-6", *Sixth Message Understanding Conference (MUC-6)*, Morgan Kaufmann, November 1995.

[2] V. Cho, B. Wuethrich and J. Zhang, "Text Processing for Classification", *Journal of Computational Intelligence in Finance*, 7 No. 2:6–22, 1999.

[3] R. Collier, "N-gram cluster identification during empirical knowledge representation generation", in *Proceedings of the Fifteenth International conference on Computational Linguistics*, pages 1054–1058, 1994.

[4] R. Collier, "Automatic template creation for information, an overview", Technical report, University of Sheffield, 1996.

[5] M. Costantino, R. J. Collingham and R. G. Morgan, "Financial Information Extraction at the University of Durham", in *Proceedings of the II Meeting of Artificial Intelligence in Accouting, Finance and Tax*, University of Huelva, Spain, September 1996.

[6] M. Costantino, R. J. Collingham and R. G. Morgan, "Information Extraction in the LOLITA System using Templates from Financial News Articles", in *Information Technology Interfaces '96*, June 1996.

[7] M. Costantino, R. J. Collingham and R. G. Morgan, "Natural Language Processing in Finance", *The Magazine of Artificial Intelligence in Finance*, 2 No. 4, 1996.

[8] M. Costantino, R. J. Collingham and R. G. Morgan, "Qualitative Information in Finance: Natural Language Processing and Information Extraction", *Neuro Ve$t Journal*, 4 No. 6, November 1996.

[9] M. COSTANTINO, R. J. COLLINGHAM AND R. G. MORGAN, “Natural Language Processing and Information Extraction: Qualitative Analysis of Financial News Articles”, in *Proceedings of the Conference on Computational Intelligence for Financial Engineering (CIFEr ’97)*, March 1997.

[10] DARPA, *Proceedings of the Third Message Understanding Conference (MUC-3)*, Morgan Kaufmann Publishers, May 1991.

[11] DARPA, *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, Morgan Kaufmann Publishers, June 1992.

[12] DARPA, *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Morgan Kaufmann Publishers, August 1993.

[13] DARPA, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Morgan Kaufmann Publishers, November 1995.

[14] R. GARIGLIANO, R. G. MORGAN AND M. H. SMITH, “The LOLITA System as a Contents Scanning Tool”, in *Avignon ’93*, 1993.

[15] R. GARIGLIANO, “Editorial”, *Natural Language Engineering*, 1, March 1995.

[16] R. GRISHAM, “Tipster Phase II Architecture Design Document (Tinman Architecture)”, 1995.

[17] G. R. KRUPKA, “SRA: Description of the SRA System as Used For MUC-6”, in *Proceedings of the Sixth Message Understanding Conference*, Morgan Kaufmann Publishers, November 1995.

[18] C. J. V. RIJSBERGEN, *Information retrieval 2nd Edition*, Butterworths, 1979.

[19] M. H. SMITH, R. GARIGLIANO AND R. G. MORGAN, “Generation in the LOLITA system: An Engineering Approach”, in *7th International NL generation Workshop*, June 1994.

[20] J. F. SOWA, *Conceptual Structures, information processing in mind and machine*, Addison-Wesley, 1984.

[21] Y. WANG AND R. GARIGLIANO, “Detection and Correction of Transfer by CAL”, in *Second International Conference on Intelligent Tutoring Systems (ITS-92)*, Springer-Verlag, June 1992.

Received: January, 1999
Revised: July, 2000
Accepted: February, 2001

Contact address:
Marco Costatino
Laboratory for Natural Language Engineering
Department of Computer Science
University of Durham
Durham
DH1 3LE
U.K.
Phone: +44 20 8932 2178
Fax: +44 20 8932 2178
e-mail: marco@advanced-finance.com

MARCO COSTANTINO studied and gained his BSc and MSc in Economics and Business Administration at the University of Trento, Italy. He later obtained his PhD in Artificial Intelligence (Natural Language Processing) applied to Finance at the Department of Computer Science, University of Durham, UK. He has published several articles in international journals in the field of Natural Language Processing and Finance. He is now working for the American Investment Bank, JP Morgan, where he is an Equity Derivatives Tr ader. He is contactable at www.advanced-finance.com where all his publications can also be found.