Financial Information Extraction Using Pre-defined and User-definable Templates in the LOLITA System

Marco Costantino, Richard G. Morgan, Russell J. Collingham

Department of Computer Science, University of Durham, Durham, U.K.

This paper addresses the issue of information extraction in the financial domain within the framework of a large Natural Language Processing system: LOLITA. The LOLITA system, Large-scale Object-based Linguistic Interactor Translator and Analyser, is a general purpose natural language processing system. Different kinds of applications have been built around the system's core. One of these is the financial information extraction application, which has been designed in close contact with experts from the financial market in order to overcome the lack of usefulness of many other systems.

Three predefined groups of templates have been designed according to the "financial activities approach": company related templates, company restructuring templates and general macroeconomic templates. In addition, the user-definable template interface allows the user to define new templates using natural language sentences.

After describing LOLITA as a general purpose base NLP system, the paper addresses the issue of how information extraction is performed within the system and how the user-definable template interface has been designed.

Keywords: Natural Language Engineering, Information Extraction, User-definable information extraction, Finance.

1. Introduction

Many natural language systems have been built to solve specific and limited tasks. LOLITA has been built with no particular application in mind.

However, the flexibility of the system allow the realisation of different kinds of applications around the system's core. One of these is the LOLITA financial information extraction application, which is described in this paper. The templates have been defined according to the "financial activities" approach which identifies three main groups of relevant financial activities in the financial market each of which is associated to a specific template. The user can also define additional templates using sentences in natural language with the user-definable template interface. The main characteristic of how information extraction is performed within the LOLITA system is that deep natural language understanding is used in order to identify the relevant information in the source articles. The work is organised as follows: in section 2 the generic information extraction task and the main systems developed in the past are described. In section 3 we describe the LOLITA system as a general purpose natural language processing system. In section 4 we describe the way in which information extraction in LOLITA is performed. In section 5 we focus on the LOLITA financial information extraction system. Finally, in section 6 we describe the natural language user-definable template interface.

2. Information extraction

The goal of *information extraction* is to extract specific kinds of information from a source document [Riloff and Lehnert, 1994]. In most systems, the output of the system will consist of *templates*, structures with a predefined number of slots.

Many of the actual systems, for example some of those developed for the MUC-5 [ARP, 1993] and MUC-6 [DAR, 1995] competition, are based on statistical and probabilistical techniques.

However, it is our belief that successful information extraction in broad domains will necessarily require deep natural language techniques rather than shallow pattern-matching or fragment parsing techniques [Lakoff et al., 1995].

Pattern-matching and statistical techniques, in fact, are usually triggered to specific tasks and constraint domains and, therefore, their portability is limited. Moving systems based on these techniques towards other domains usually means major changes to the algorithms and patterns employed. Differently, deep natural language processing is not triggered to any specific domain and, therefore, allows the maximum degree of portability towards other domains. The broad financial domain can be considered an example of this. Pattern-matching and statistical techniques would be able to perform successfully in specific situations, but would be unable to consider all possible cases. Deep natural language processing is instead independent from specific cases and, therefore, represents the best choice for such broad domains. Information extraction within the LOLITA system follows this approach.

2.1. Information extraction in finance

The goal of information extraction applied to finance is, as within other domains, to extract relevant information from a text producing an output usually consisting of a template of the original text. The information extracted from the source texts can be used by the financial operators to support their decision making process and to analyse the effect of news on price behaviour [Costantino et al., 1996b].

One of the outstanding properties of the texts that make up financial articles, as opposed to general free-text, is the presence of a considerable high number of metaphors. A randomly selected text from the *Financial Times* of October, the 21th 1992 showed 11 metaphorical expressions in the first sentence, which was 37 words long.

Financial Times 21 Oct 92 London Stock Exchange: Equity futures and options trading

The *German Bundesbank's* decision *to move to* a variable repo rate, *leading to strong speculation* of further *cuts* in UK base rates, *enlivened a dull derivatives sector* and *sent* Footsie futures *moving sharply ahead*, writes Joel Kibazo.

Very few financial information extraction systems have been realised in the past. One of the few systems is ATRANS [Lytinen and Gershman, 1986], a system for extracting information from telex messages regarding money transfers between banks. The system was based on the script-frames approach and it has been successful mainly because of the extremely limited domain and the limited information to be extracted. The system that competed in the MUC-5 competition [ARP, 1993] were also able to perform the extraction of information from financial articles. However, these systems were only able to extract information regarding *Joint Ventures* and, thus, work in an extremely restricted subset of the financial domain.

3. The LOLITA system

LOLITA (Large-scale Object-based Linguistic Interactor Translator and Analyser) is a *general purpose* natural language processing system and has been under development at the University of Durham for the last eight years [Garigliano et al., 1993].

The system has been built with no particular application in mind. This means that different kinds of applications can be easily built around the original system's core. The approach taken for designing and implementing the system follows the lines of natural language engineering rather than those of traditional computational linguistics. The NLE approach emphasises the following aspects of engineering that should be considered when building a NL system [Garigliano, 1995].

Scale: the size of NLE systems must be sufficient for realistic large-scale application. Properties such as as the vocabulary size, grammar coverage and the number of word senses are critical.

Feasibility: this aspect concerns ensuring that constraints on the running of the system are acceptable. For example. hardware requirements should not be too great and execution speed must be adequate. Feasibility incorporates making the system and its components efficient.

Robustness: robustness is a critical aspect of large-scale systems. Robustness concerns not only the linguistic scope of the system but how it deals with input which falls outside of this

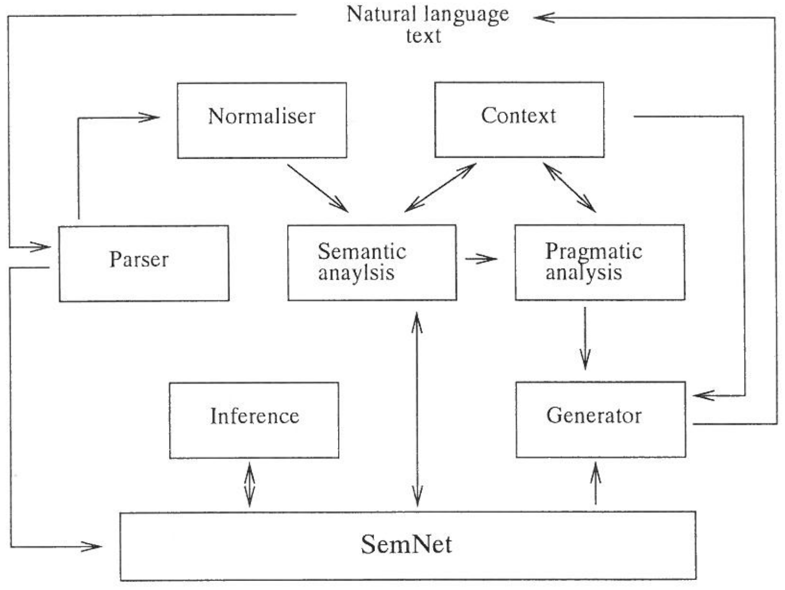

Fig. 1. The LOLITA system core.

scope. At the very least, it should be able to carry on and try its best to cope with the conditions it is working under.

Maintainability: maintainability is a measure of how useful the system is over a long period of time. There are four different aspects that can be referred to the term: *corrective maintenance of software repair, enhancement, perfective maintenance, preventive maintenance.*

Usability: the system must satisfy a need, i.e. there must be a set of users in the market who can benefit from using the system [Garigliano, 1995].

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 [Sowa, 1984]. 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 connected with arcs. The nodes represent entities (*loss*), events (*The company made losses*) and relations (*A company ISA business*). The knowledge is stored in the network by using control variables. Control variables are the essential information stored at each node, there are about 50 different control variables. Knowledge is represented in the Semantic Network according to the connectivity between the nodes and arcs. 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 losses*), generalization (*shares are a form of investment*), specialization (*one form of investment are shares*) etc.

- **Type.** This control values are 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 [Garigliano et al., 1993].

- **Family.** This control groups nodes into the semantic "families", eg. living, animal, human, man-made, abstract, location, organisation, human-organisation etc. [Garigliano et al., 1993].

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 jerarchical 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;
- 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 [Smith et al., 1994]. 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 [Wang and Garigliano, 1992], query application and contents scanning [Garigliano et al., 1993].

4. Information extraction in the LOLITA system

The way in which the information extraction application works is quite straightforward. The articles are firstly processed by the LOLITA system and stored in the semantic network. The task of the template application is then to search for the information needed in the semantic network. Finally, the output is produced either using the *generator* or referring to *fragments* of the source document.

The template application, thus, interacts with the LOLITA system core retrieving data from the *semantic network* by using the *inference system* and producing output in English either by using the *generator* or by referring to the original text.

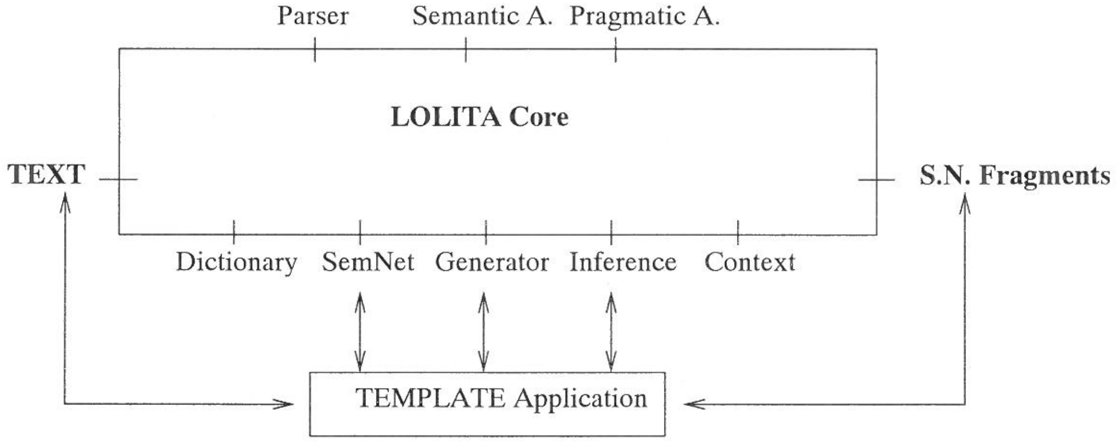

Fig. 2. Information extraction using LOLITA.

The most important characteristic of the way in which information extraction is performed within the LOLITA system is the fact that the templates are built only using deep general purpose natural language techniques, rather than shallow pattern matching techniques. This approach has the advantage of allowing the maximum degree of portability towards new domains, since the template application is built upon the domain independent LOLITA core.

A template is defined in LOLITA as an Haskell data-type comprising a predefined set of slots with associated fill-in rules that direct the search for appropriate information in the semantic network. There are currently five different type of slots available, which differ according to the way in which the slot is filled.

Concept Slot. This slot represents the generic LOLITA templates slot. The rule associated with the slots identifies the relevant concept in the semantic network which is passed to the generator obtaining the corresponding English text.

String Slot. This slot produces the output directly from a given string and not using the English generator. It is mainly useful to produce a slot with a predefined number of alternatives which may not be present in the original text, e.g. **Type of Takeover:** *FRIENDLY / HOSTILE*.

User-Defined Slot. In this case, the output is produced by the English generator. The slot fill-in rules are retrieved from the semantic network, allowing the creation of user-defined templates or rules by updating portions of semantic network. This kind of slot is used by the NL user-definable template interface (see section 6).

Text Reference Slot. The output is produced using fragments of the original text where possible and the generator if a semantic network's concept doesn't correspond to any fragments of the original text (e.g. when using inference functions). The slot is used when the exact copy of the original text is needed (e.g. in the MUC-6 templates).

Template Reference Slot. This slot is used to create a link to another template. The output of the slot will consist of a pointer to the new template. The *template reference slots* provide the basic mechanism for handling *hyper-templates* in LOLITA which are widely used in the MUC-6 scenario templates [Morgan et al., 1995].

The slot fill-in rules are used for locating the relevant information for a particular slot in the semantic network. The rules can be built by checking the control variables associated with the nodes or by using the inference functions available in the core system in case the information is not directly stored in the semantic network but has to be inferred. The rules can be used to retrieve any kind of nodes from the semantic network (cf. section 3). However, template slots will usually be filled with *entities* or *events*.

For example, the identification of a company is performed by searching in the semantic network for all the new nodes that belong to the families *organisation* or *human organisation*. The semantic network is in fact updated with all the new nodes created by the semantic analysis of the source documents during the core analysis.

If the information to be located is an *event*, the searching process will involve the identification, through inference functions, of the subjects, objects or target events of the event itself.

Three different kinds of templates are currently available in the LOLITA system, the *Event-based templates*, the *Summary templates* and the *Hyper templates*.

4.1. Event-based Templates

Event-based templates are structures where it is possible to identify a clear underlying top-level event to which all the information of the template's slots can be referred to [Garigliano et al., 1993].

For example, the *takeover* of a company by another company can be considered a suitable event for building an event-based *takeover* template. In fact, all the information regarding the takeover: *amount*, *type of takeover*, *company target*, *company predator* etc. can be associated to the "parent" *takeover event*.

More than one event-based template can be identified in a source document, according to

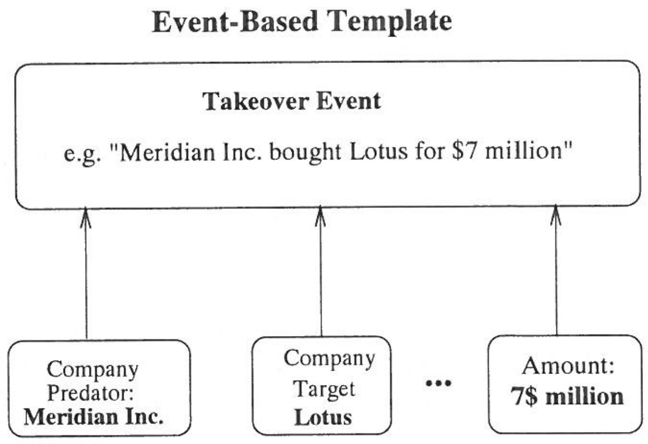
Event-Based Template
Takeover Event
e.g. "Meridian Inc. bought Lotus for $7 million"
Company Predator: Meridian Inc.
Company Target Lotus
...
Amount: 7$ million

Fig. 3. Event-based templates.

the number of relevant top-level events that are generated by the semantic analysis of an article.

Once the template's *parent event* has been identified each slot is filled by searching in the semantic network for the relevant information according to the slots' fill-in rules and starting from the node of the *parent event.*

4.2. Summary-templates

Summary templates represent structures for which it is not possible to identify a *parent event.* A summary template is basically a collection of objects stored in different slots which may not directly refer to the same concept or relate to each other. For example, a summary template can be composed of the following slots *personal names, organisations, locations, temporal, acronyms, monetary values, descriptions, animates, inanimates* (figure 4).

Summary templates are built differently from event-based templates. In fact, the slots' fill-in rules are applied to the list of all the nodes generated by the semantic analysis of the source document since a *parent-event* doesn't exist.

4.3. Hyper-Templates

Hyper-templates are structures whose slots can refer to other templates, thus creating a linked *chain* of templates. For example, in the *takeover template* (described later in this section) the slots *company predator* and *company target* could potentially be linked to an *organisation* template which would include detailed information regarding the company. The hyper-template mechanism is potentially usable for linking different kinds of information, not necessarily extracted from the source text, such as company databases or historical share prices. Hyper-templates have been used for implementing the MUC-6 scenario dependent templates (a complete description of the implementation of the templates can be found in [Morgan et al., 1995]).

5. Financial information extraction in LOLITA

Four main aspects have to be defined for designing a financial information extraction sys-

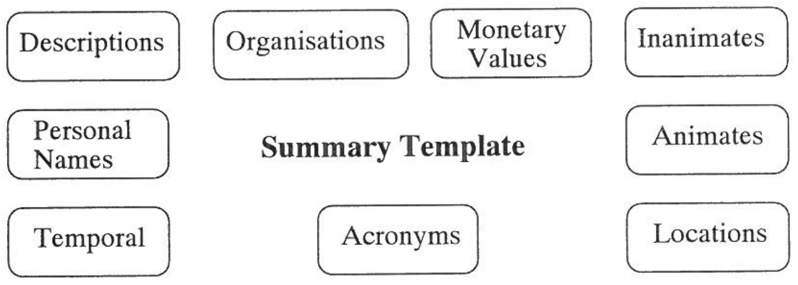

Fig. 4. Summary Templates.

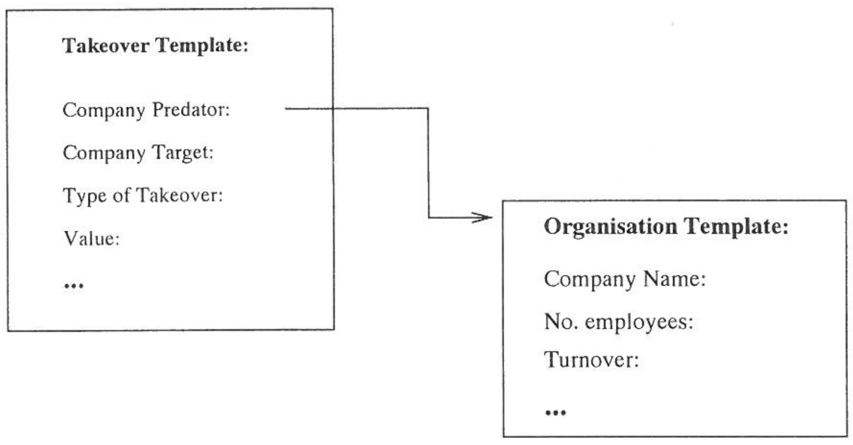

Fig. 5. Hyper Templates.

tem: the kind of source articles (from on-line services or from newspapers or magazines), the information to be extracted, the output of the system (summaries or templates) and the interface to the user. Assuming that the output of the system consists of templates, the most important decision is the kind of information to be extracted.

Financial articles represent an extremely wide domain, including different kinds of news: financial, economical, political etc (cf. section 2.1). Therefore, the identification of a unique template able to summarise all the possible financial articles is extremely difficult, if not impossible. The best solution is thus to design more than one template.

Financial information extraction in the LOLITA system is based on the *financial activities approach* [Costantino et al., 1996a]. A *financial activity* is here defined as one potentially able to influence the decisions of the players in the market (brokers, investors, analysts etc.) regarding securities issued by companies. A finite number of relevant *financial activities* are identifiable in the financial market and can be grouped into three different categories.

- ***Company related*** activities which are those related to the "life" of the company, changes in its status, in the ownership of the company, the number and ownership of its shares etc. This group includes the following financial activities: merger, takeover, flotation, new issue (of shares, bonds etc.), privatisation, market movement, bankruptcy, broker's recommendations, taking a stake, dividend announcement, overseas listing, profits/sales forecasts, profits/sales results, directors' dealings, legal action, investigation.

- ***Company restructuring*** activities. This activities are related to changes in the productive structure of the company and include: new product, joint venture, staff changes, new factory.

- ***General macroeconomics*** activities, which include general macroeconomics news that can affect the prices of the shares quoted in the stock exchange and comprises: interest rate movements, currency movements, general macroeconomics data (inflation, unemployment, trade deficit).

A specific template is associated to each of the *financial activities* and comprises a predefined set of slot. In figure 6 the definition of the takeover template in BNF notation is shown.

The implementation of the financial application within the LOLITA system is straightforward. The documents are firstly processed by the system. The analysis takes into account *prototypical* information and the *topic* which has been set for the financial application. *Prototypical information* are used by the pragmatic analysis module to restrict the kind of entities that can be used within a particular event according to the appropriate action and for the disambiguation of word meanings. One of the relevant *prototypes* identified for the financial templates is the *acquisition* prototype. The subject of the acquisition can be either a *person* or a *company*, while the object can be a *company*. The action can be

```
Company Target:            COMPANY_NAME
Company Predator:          COMPANY_NAME
Type of takeover:          {FRIENDLY, HOSTILE}
Value:                     [NUMBER] & CURRENCY_UNIT
Bank adviser predator:     BANK_ADVISER
Bank adviser target:       BANK_ADVISER
Expiry date:               DATE
Attribution:               ATTRIBUTION:
Current Stake predator:    {PERCENTAGE, SHARES, AMOUNT} [NUMBER]
Denial:                    DENIAL

COMPANY_NAME, CURRENCY_UNIT, BANK_ADVISER, ATTRIBUTION, DENIAL: String
DATE: [[0-31][0-12][00-99]]
PERCENTAGE, AMOUNT: Value
```

Fig. 6. The Takeover template (BNF notation)

acquire, *buy* or any other verbs with appropriate meanings.

The *topic* of an article can be defined as the *theme* or the *subject of discourse* of the article. For example, it is likely that the meaning of the verb *to buy* in a financial article will be:

```
buy: To take., To purchase, To buy. ->
To acquire;
```

rather than:

```
buy: To corrupt., To bribe. -> To pay.
```

The information stored in the *topic* is used by the pragmatic analysis module to disambiguate the meanings of words. The *topic* for the financial application includes the appropriate meanings of words and concepts one would expect in a financial context, for example *buy*, *takeover*, etc.

Once the analysis of the source article has been completed and the new knowledge stored in the semantic network, the financial application will look for any event which match the main-events corresponding to the financial activities. For example, a *takeover* is identified by the application looking for any *takeover* event in the semantic network. The *takeover* event is defined as follows:

- the event that can be generalised to the concept of *takeover*, *acquisition* or *purchase*. For example "The acquisition of X by Y", where X and Y are companies;
- the event that has a *takeover-action*, e.g. *to buy*, *to take-over*, *to purchase*, etc. and the object is a company. For example: "X has acquired Y for 100 million dollars", where Y is a company.

Similarly, the conditions of all the other financial activities (events) are checked and the full list of the financial activities found is shown to the user. Subsequently, the slots are filled using the associated slot-rules. For example, the *company predator* of the *takeover* template will be identified as the subject of the takeover event, while the *company target* is the object.

Four kinds of slots: concept slots, string slots, text reference slots and template reference slots (cf. section 4) can be used to define the financial templates. For example, the *company target* slot of the *takeover* template is defined as *concept slot*, while the slot *type of takeover* is defined as *string slot*, since it can only assume values *friendly* or *hostile* which may not be explicitly stated in the source document and, therefore, must be inferred by the system. The *hyper-template* mechanism can be used in a large number of cases. For example, in the *takeover* template, the slots *company predator*, *company target*, *bank adviser predator* and *bank adviser target* can potentially be linked to other templates or other sources of information, such as company databases. In the same way, the *market movement* template could potentially be linked to a historical shares price database.

An important chapter in the development of the financial application is the domain-specific knowledge. In fact, financial articles are based on highly technical and specific knowledge and

Source article from the Financial Times:

Filofax group announced yesterday it will acquire Drakes Office Systems with 3 million dollars from its founder, Mr. Tom Drake. Initial consideration comprises a mixture of cash and the issue of 1m ordinary shares. Drake claims to be the UK market leader in Wire-O bound carbonless duplicate message books. Its Ring Back brand forms a range of telephone message and similar business forms with a dominant market share. In 1992, Drakes made gross profits of Pounds 727,000 on sales of Pounds 1.7m.

Template produced:

```
Template: TAKEOVER
          COMPANY_TARGET:    Drakes Office Systems.
          COMPANY_PREDATOR: Filofax Group
          TYPE_TAKEOVER:    FRIENDLY
          VALUE:            3 million dollars.
          ATTRIBUTION:      Filofax Group.
```

Fig. 7. An example of a LOLITA-produce takeover template

lexicon. Sentences which would normally have a certain meaning in a normal text might present a totally different one in a financial context. We can think of domain-specific knowledge in the context of a large Natural Language Processing System such as LOLITA as *semantic and pragmatic rules* which are used by the system to correctly *understand* the source text and choose the right meaning of a particular word in the financial context. As far as the takeover template is concerned, various domain-specific rules have been identified:

- if X takes full control of Y, this implies a takeover;
- X buys a majority stake in Y, this implies a takeover;
- X buys a 51 (or over) per cent stake in Y, this implies a takeover;
- X pays M for Y and Y is a company, this implies the takeover of Y by X.

At present, the financial application is only partially implemented within the LOLITA system, only the first group of financial activities (company related activities) has been partially coded in the system.

One of the disadvantages of the way in which information extraction is currently performed within the LOLITA system is the fact that only deep natural language techniques are used. Therefore, the performance in terms of speed can be, in particular situations, penalised in comparison to systems based on pure pattern-matching or fragment-parsing techniques.

6. The NL user-definable template interface

The templates defined according to the *financial activities* approach should, in our view, represent the information which is relevant for the financial operator's investment decision-making process. However, the financial application allows the user to define additional templates using the *user-definable* template interface. This allows the maximum degree of flexibility for the user. A template such as the takeover template shown in figure 7 is defined in the LOLITA system by the following four 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.

In the same way, the user-definable template interface requires the user to input the definition of these four elements. An experiment conducted on a number of potential users of the system showed that allowing the user to define these four elements using complete free natural language sentences leads to a high percentage of ambiguities for both the user and the system. Therefore, a set of *formal elements* have been introduced. The user can define new templates using sentences in natural language which make use of a set of formal-elements designed to reduce the amount of possible ambiguities in the templates definitions but does not reduce the user's expression power. Three different *formal elements* have been introduced:

- the **name of the template** which must begin with the letters "*T=*". The name of the template can be used in the definitions of the slots to refer to the template as a whole, for example, the slot *S=VALUE=TAKEOVER* in the takeover template is defined as "*the cost of the T=TAKEOVER*" (figure 8).
- the **variables** which can be defined by the user (beginning with the letters "*V=*") to identify the elements of the main-events which will be later used in the definition of the slot-rules. For example, in the definition of the takeover template, the user can define the variable "*V=COMPANY1 is a company*" which is used to identify the company predator and, therefore, appears in both the *main-events* and the slot-definitions (figure 8).
- the **slot-names** which can be used in the definition of other slot rules to refer to the information contained in the previous slot. For example, the slot *S=ATTRIBUTION* of the takeover template (figure 8) refers to the other slots.

6.1. Processing the user-definitions

Once the definitions of the templates has been entered by the user, the user-definable interface will process them and produce an appropriate representation of the template in the semantic network, which will be used by the inference engine to fill the templates.

A node corresponding to the name of the template, for example "*T=TAKEOVER*" is firstly created. The node is associated to the control value "templvariable: yes" and is given rank "rank_Individual". The node is linked to the concept "typeless" using a "is_a" link processing the following sentence:

```
T=TEMPLATE-NAME is typeless.
```

The control value "*templvariable*" has been introduced to identify in the semantic network concepts which correspond to the *formal elements* of the template user interface and can assume the following values:

```
templvariable:  no
templvariable:  yes
```

where the default value for concepts which do not correspond to any *formal elements* of the template user-interface is "*no*".

The definitions of the variables are then processed. Since the definitions of the variables are entered using full sentences, they can be directly processed by the system without any modification or normalization. For example, the definition of the variables:

```
V=COMPANY1 is a company
```

and

```
V=COMPANY2 is a company
```

are processed by the system obtaining an event where the variable "*V=COMPANY1*" and "*V=COMPANY2*" are the subject of two different events with action "*is_a*" and object the generic concept of *company*.

After the definition of the variable has been processed, the control "*templvariable: yes*" is assigned to the variable, to identify it as a *formal element* in the semantic network. The variable is also marked with rank "*rank_Individual*", while the family is assigned automatically by the system during the analysis. The representation for the variable "*V=COMPANY1 is a company*" is shown in figure 9.

```
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  T=TAKEOVER is not hostile.

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-TAKEOVER
                        or denied EXPIRY-DATE.
```

Fig. 8. The takeover template as defined for the template user-interface.

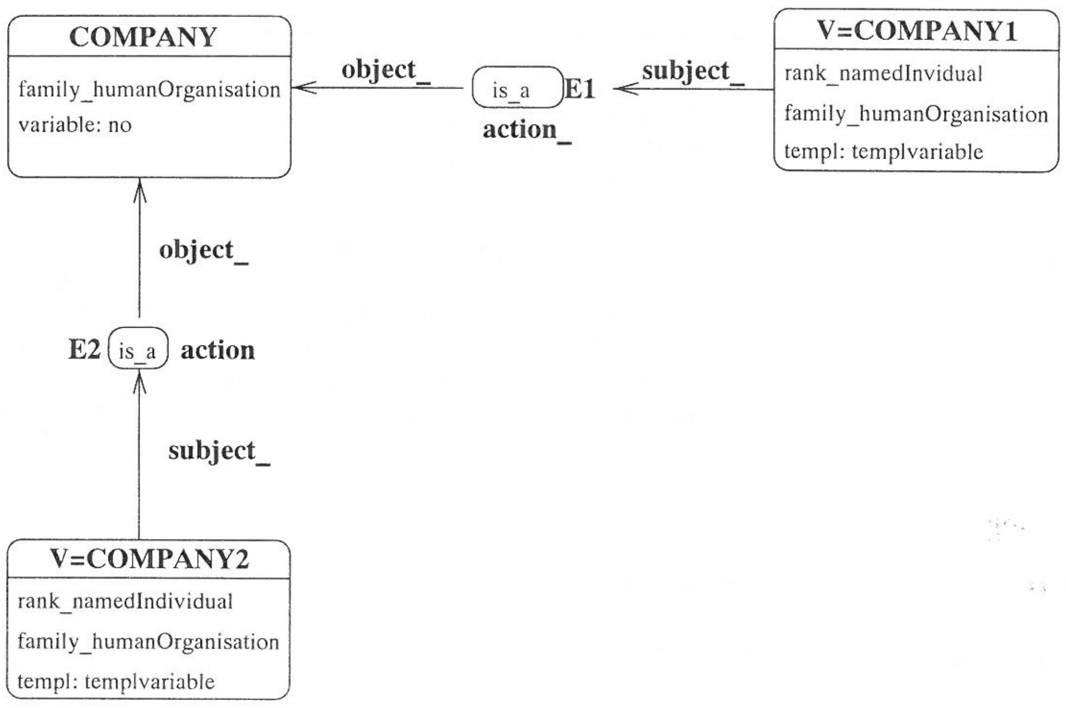

Fig. 9. The processing of a user-defined variable

The template condition definitions are subsequently analyzed by the system. Similarly to the definitions of variables, the main-events are entered by the user in the form of full sentences. The definitions are therefore processed without any modification or normalization. If the main event contains variables, these are directly linked to the variables definition already processed by the system. The definitions are marked as *questions* by the interface since they will represent the questions which will be passed to the inference engine.

The analysis of a main-event such as:

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

will produce a new event with action *acquire*, subject *V=COMPANY1* and object *V=COMPANY2* and *status_* "*wh_question*" (figure 10).

Multiple main-events corresponding to the same template condition are similarly processed and can refer to the same variables.

Each of the noderef corresponding to the main-event is stored in the Haskell data structure *TemplateRule* which is defined as follows:

```
> data TemplateRule = TemNetUserDefin
[MainEvent]
```

The final step is the processing of *slot-names* and *slot-rules*. Slot names are stored in the *Template* data structure. The definition of the slot rules are instead processed by the parser as Noun-Phrases and also stored in the template data structure.

6.2. Filling the templates using the inference system

The way in which templates are filled by the user-definable template interface is rather different from how the predefined financial templates are handled.

First of all, no code describing the template rules are available in the system. The definitions and rules for the predefined financial templates are coded directly within the system, as part of the source code. User-defined templates, instead, are filled by the system using the *inference engine* 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.

The *inference engine* identifies entities and events which satisfy the template rules stored in the semantic network corresponding to the *variables* the template *main-events* and the *slot-rules*.

Inference and the main-events

The first condition which is analysed by the inference system is the set of *main-events* stored in the TemNetUserDefin data structure. Each of the main-events, which have already been marked as *wh_question* by the interface (e.g. the event in figure 10), are passed as question to the inference engine.

The inference functions will look for any events which match the given main-event and, in this case, the event will be a candidate event for filling-in the templates slots. The inference functions will also identify the relevant nodes which can be matched against the variables which will be used for filling slots which refer to variables.

For example, for the main-event shown in figure 10:

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

the inference engine will recognize that an event such as:

```
FIAT acquired RENAULT
```

is a relevant one, because of the fact that the action is the same and the subject and object can be matched against the variables V=COMPANY1 and V=COMPANY2. In figure 11 the representation of the main-event and the candidate event "FIAT bought RENAULT" is shown. The inference system tries to match each of the components of the candidate event onto the main-event.

The inference engine 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 engine for searching for concepts which match the slot rules.

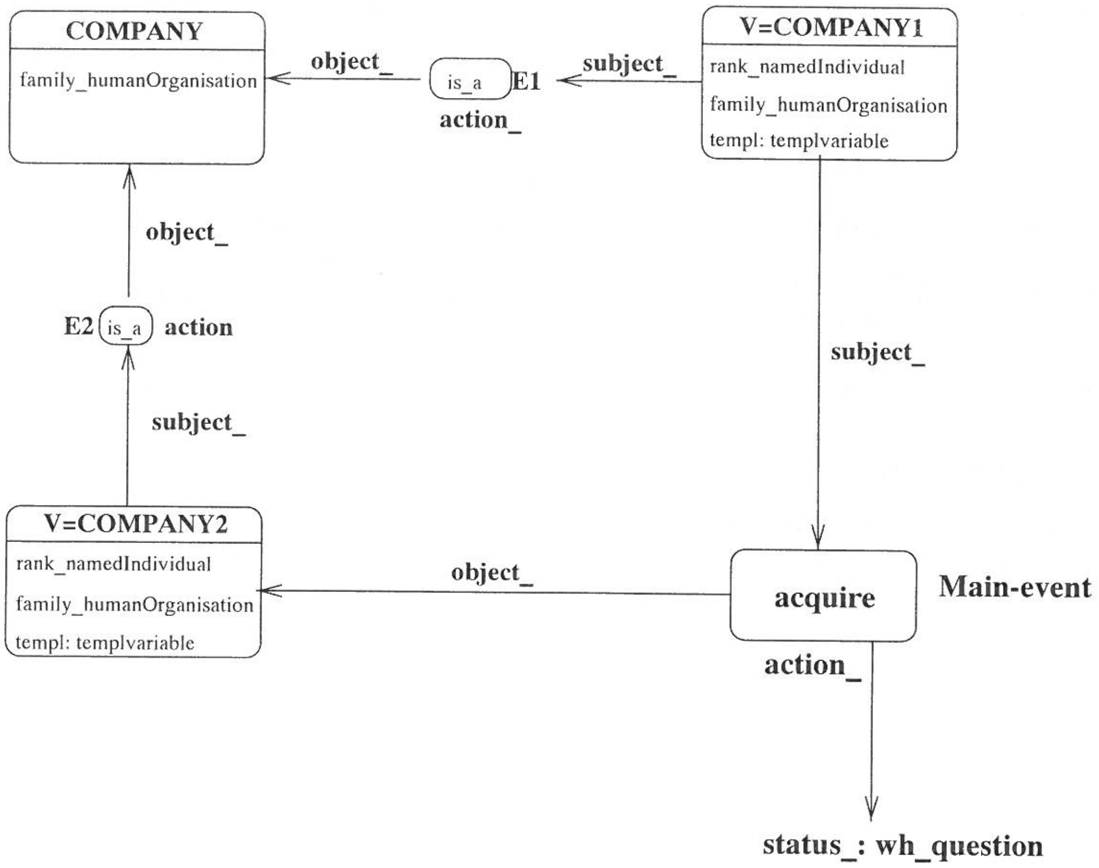

Fig. 10. The processing of the main-event

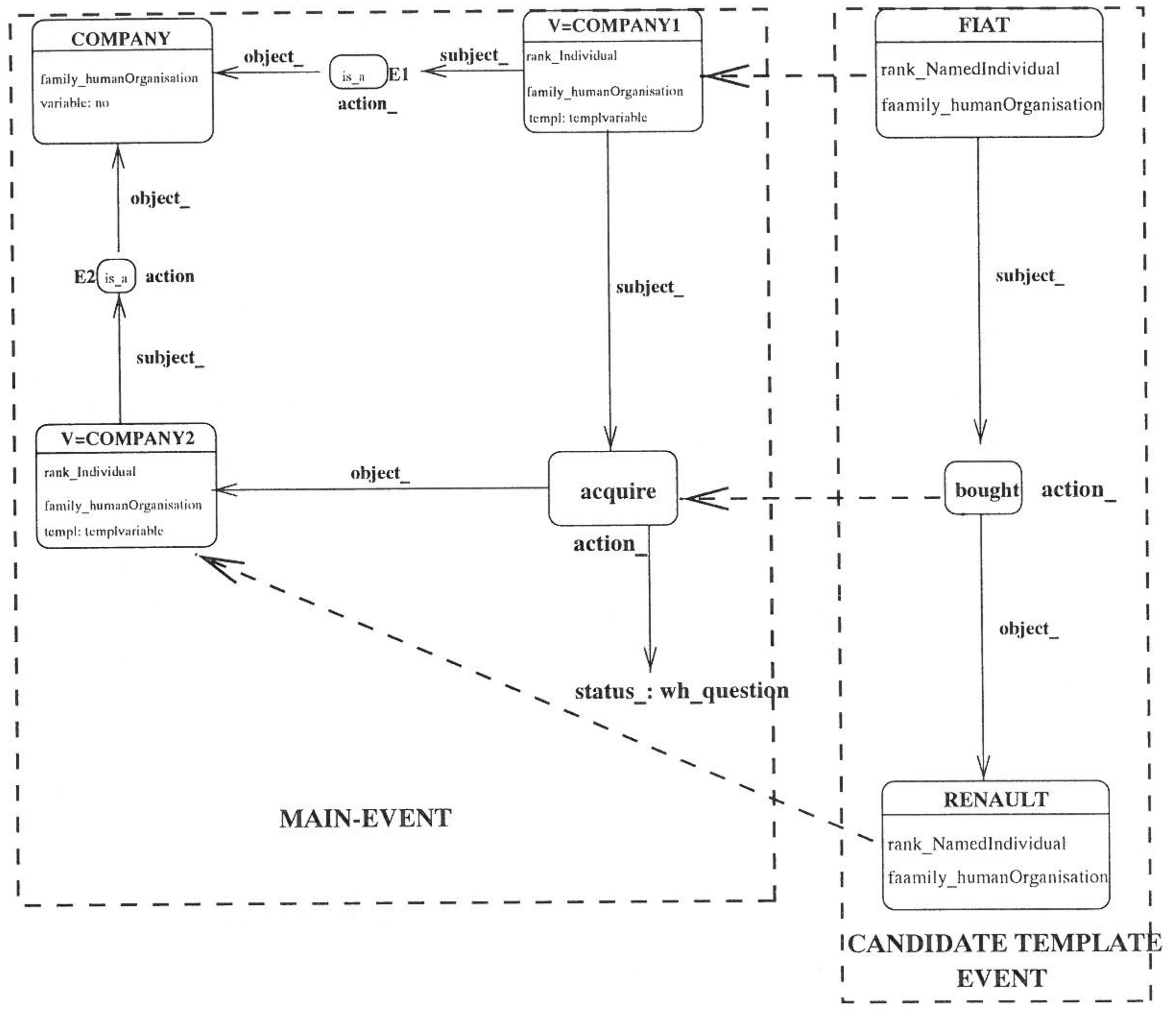

Fig. 11 Identification of candidate main-events by the inference system.

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 is 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-TAKEOVER:        V=VALUE
  ```

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

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

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

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

The template user-definable interface is currently under development. At present, the template definitions are correctly processed and stored in the semantic network, while the inference engine is connected to the application. However, the inference rules specific to the user-definable interface have not yet implemented.

7. Conclusions

In this paper we have tried to give an overview of how financial information extraction is performed in the LOLITA system. Unlike many other information extraction systems, the emphasis is on full natural language processing of the text.

In a paper of this size it is impossible to provide details of every aspect of a large NL system such as LOLITA. Because of the commercial value of the LOLITA project, the system is not publicly available. However, we are keen to give demonstrations of the system and serious enquiries should be addressed to the authors.

References

[ARP, 1993] ARPA, Proceedings of the Fifth Message Understanding Conference, Morgan Kaufmann Publishers, August 1993.

[Costantino et al., 1996a] 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.

[Costantino et al., 1996b] M. COSTANTINO, R. J. COLLINGHAM, AND R. G. MORGAN, "Qualitative Information in Finance: Natural Language Processing and Information Extraction", NeuroVe$t Journal, November 1996.

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

[Garigliano et al., 1993] R. GARIGLIANO, R.G. MORGAN, AND M.H. SMITH, "The LOLITA System as a Contents Scanning Tool", in Avignon '93, 1993.

[Garigliano, 1995] R. GARIGLIANO, "Editorial", Natural Language Engineering, 1, March 1995.

[Krupta, 1995] G. R. KRUPTA, "SRA: Description of the SRA System as Used for MUC–6", in Proceedings of the Sixth Message Understanding Conference, Morgan Kaufmann Publishers, November 1995.

[Lakoff et al., 1995] G. LAKOFF, J. ESPENSON, AND A. SCHWARTZ, Master metaphor list, Berkeley, California, 1995, http://cogsci.berkeley,edy/.

[Lytinen and Gershman, 1986] S. L. LYTINEN AND A. GERSHMAN, "ATRANS: Automatic Processing of Money Transfer Messages", in M. Kaufmann, editor, 9th International Joint Conference on Artificial Intelligence, pages 821-825, 1986.

[Morgan et al., 1995] R. MORGAN, R. GARIGLIANO, P. CALLAGHAN, S. PORIA, M. SMITH, A. URBANOWICZ, R. COLLINGHAM, M. COSTANTINO, C. COOPER AND THE LOLITA GROUP, "University of Durham: Description of the LOLITA System as used in MUC-6", in Sixth Messages Understanding Conference (MUC-6), Morgan Kaufmann, November 1995.

[Riloff and Lehnert, 1994] E. RILOFF AND W. LEHNERT, "Information Exttraction as a Basis for High-Precision Text Classification", ACM Transactions on Information Systems, 12 No.3:296 333, 1994.

[Smith et al., 1994] 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.

[Sowa, 1984] J. F. SOWA, Conceptual structures, information processing in mind and machine, Addison–Wesley, 1984.

[Wang and Garigliano, 1992] 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: October, 1996
Accepted: March, 1997

Contact address:
Marco Costantino, Richard G. Morgan, Russell J. Collingham
Department of Computer Science
University of Durham
Durham, DH1 3LE, U.K.
Tel +44 191 374 2549
Fax +44 191 374 2560
e-mail: marco.costantino@durham.ac.uk

MARCO COSTANTINO holds a "dottore in Economia e Commercio" (Economics and Business Administration) from the University of Trento (Italy). He is currently a PhD. student at University of Durham, department of computer science, Laboratory for Natural Language Engineering. His field of research is Natural Language Engineering in Finance.

RICHARD G. MORGAN holds a Bsc and PhD. in computer science. He is currently lecturer at the University of Durham, Department of computer science. His main research areas are Natural Language Engineering, Functional Programming and Information Extraction.

RUSSELL J. COLLINGHAM holds a Bsc and PhD in computer science. He is currently lecturer at the University of Durham, Department of computer science. His main research areas are Speech Recognition, integrating Speech and Natural Language Systems and Information Extraction.