PARALLEL INFORMATION EXTRACTION SYSTEMS FOR MULTILINGUAL INFORMATION ACCESS

LUCA DINI
CELI-Centro per l'Elaborazione del Linguaggio e dell'Informazione
Torino, Italy

1. Introduction

Information Extraction[1] is mostly a monolingual technology. This feature is evident when considering the most important worldwide initiative focusing on Information Extraction, i.e. the Message Understanding Conference. In this competition English has always been the unique target language, with the exception of MUC-6 (MET-1) where Spanish and Chinese were considered as well. Even the introduction of these languages, however, has not significantly changed the monolingual status of the competition, as all the systems were assumed to be completely independent. This situation contrasts with Information Retrieval where, almost since the beginning ([1]) multilinguality was one of the important research tracks, as attested by the introduction of a Cross-Language IR track in TREC-6.

In a sense, this situation is induced by the very nature of the technology. Indeed, there is a commonly agreed assumption that IE should produce templates or database records from natural language texts. So, at the end of the processing chain, there is not a human with his/her own language, but a database or a repository of templates. On the contrary, the typical processing chain for Information Retrieval is from human to human (query processing and presentation of the results), passing through information mainly encoded by humans (the document base). Human involvement is what makes multilinguality an issue.

Besides the lack of human involvement on the final edge of the processing chain, the other reason why so little attention was paid to multilinguality in IE, is probably that there is a clear cut way to switch a monolingual system into a multilingual one: as soon as two systems dealing with two different languages share the same template system, the sum of the two give raise to a multilingual system. This shifts the attention from the processing phase to the one of template design. In particular, for a system of templates to be language independent, it must assume that it contains a fixed number of predicates ranging only over named entities, numbers and other predicates. More

[1] I would like to thank Andrea Bolioli, Vittorio Di Tomaso, Gregor Erbach and Hans Uszkoreit for important support and advices.

formally, assuming that templates are described in terms of typed feature structures, the following definition could be given:

A language independent system of templates is described by
1. A finite set of predicates, defined by the types of the description language.
2. A finite set of attributes
3. A possibly non finite set of values represented by types, integers and strings. Whereas a string appears as a value of an attribute, it has to denote a named entity, i.e its linguistic realization must not change across languages.

This is obviously only one way of formulating constraints on language independent templates. For instance, as an alternative, it is perfectly conceivable that strings rather than types represent predicates. In this case, it is not necessary that all strings denote named entities, but it has still to hold true that the number of strings not denoting named entities has to be finite. In any case, the basic idea is still the same: templates should not contain fields that are string of text "unknown" to the system.

For instance the application of IUTA described in ([2]) *is* language dependent. In that case the task was to analyze job announcements and to store them into a database. However, being the possible range of job qualifications quite wide, the choice was made of not listing all of them internally to the system. For instance, we decided that, in the string *in qualità di X* ("as a X"), the string value of X should be the filler of one of the field **role** of the DB record. This was an easy choice, which simplified the processing on a monolingual side, but had the effect of making the templates intrinsically language dependent. Indeed, the filler of the **role** field is now a fragment of Italian language, with no correspondence in the abstract metalanguage of templates.[2]

Notice that the strategy of handling multilinguality in IE by adopting language independent templates is not the only possibility. In machine translation there are two basic approaches, i.e. the interlingua based approach, and the transfer based approach. What we hinted at so far is the analogous of the interlingua approach in the field of IE. However, there might be cases where the other approach is sensitive as well. In this case different systems would "talk together" by resorting to special domain tailored

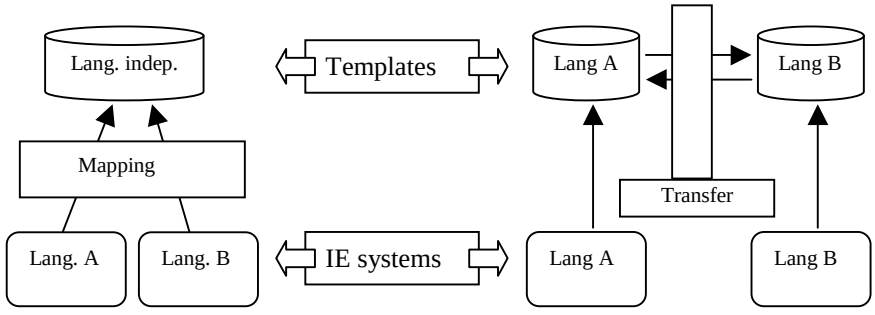

Figure 1: Different views of Multilingual IE

[2] For another example of language dependent templates see for example the description of the MUC-4 templates as emerging from [14].)

bilingual dictionaries and transfer components (cf. Fig. 1). Since the application we are going to describe does not rely on the use of multilingual dictionaries, from now on we will concentrate our attention uniquely on the former strategy.

2. IE and Natural Language Generation

As we mentioned, multilinguality becomes an issue as long as humans speaking two different languages are involved in the processing flow. Thus, two systems relying on the same template design do not, in principle, describe a multilingual system. However, as soon as the DB of templates is accessed by non-expert humans (i.e. persons unable to deal with the internal representations of the DB) the problem of displaying information in the language of the user raises its head again.

The obvious answer to such a problem is natural language generation. Templates and sets of templates represent by themselves an ideal input to natural language generation (compare for instance the description of the MUC templates with the input of the system described in [3]). In the simplest case they can feed a system of canned text generation. In more complex cases some variations on the syntactic structure might be required. Finally, the most interesting situation is raised by cases when the system has to deal with *sets* of templates that jointly provide an answer to the query posted by the user. Then there is room for a full-fledged system, exploiting sophisticate text planning strategies.

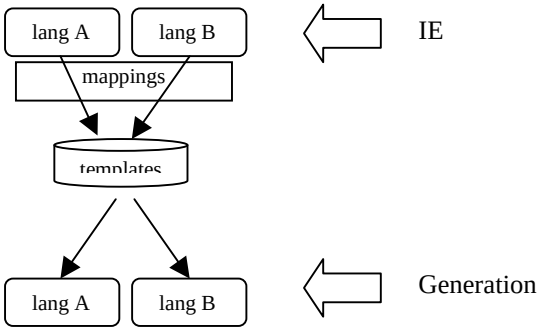

Figure 2: IE/NL-Generation

Notice that the exploitation of language independent template design and natural language generation is likely to solve some problems of multilingual DB maintenance, irrespective of an underlying IE system. Indeed, it is not uncommon to see Internet accessed DBs that are maintained in as many versions as the languages in which it is supposed to provide an output. In most cases, this effort of maintenance is due to the fact that the DB contains also small pieces of natural language information necessary to the visualization of the retrieved DB contents. By assuming a language independent template design coupled with natural language generation, such an effort of maintaining parallel DBs can be drastically reduced. Moreover, simple IE systems can provide a substantial help in bridging the language aware source DB into the new one,

where natural language strings have been replace by more structured language independent information.

3. MIETTA's Overview

The MIETTA[3] project (Multilingual Information Extraction for Tourism and Travel Assistance, LE4-8343) is aimed at building a prototype implementing the approach to multilingual IE described so far. The goal of the project is to provide Web users access to touristic information in their own languages. Four languages are considered, i.e. English, Finnish, German and Italian. Content providers such as municipalities and state/regional administrations provide the repository of documents.

As a final outcome, MIETTA will enable the end user to use a standard web browser (e.g. Netscape) as his/her interface to access "MIETTA enhanced tourist servers". From here, the user is able to express requests for documents by means of natural language queries. If the query expressed by the user is over- or underspecified from the system's point of view (or if it is expressed in such a way that the IE system for that language cannot get any useful information out of it), the system can ask the user for more precise information to constrain further retrieval. This strategy will be coupled with a search strategy based on advanced information retrieval techniques.

The set of documents retrieved for a query will be presented to the user in an informative way. The ranking of the documents will be determined either by the user profile (if any) or by standard ranking criteria. Moreover, for every document it will be possible to produce, through natural language generation, a short summary of the information it contains, presented in the user language.

3.1 THE ARCHITECTURE

The core of the MIETTA architecture is represented by four "aligned" systems for IE and NL-generation targeting English, Finnish, German and Italian. University of Helsinki will provide the modules for Finnish, DFKI the ones for German and CELI the ones for Italian. As for English, the MIETTA consortium will provide an adequate processing chain simply by putting together modules that are already available to the single participants.

From the point of view of input/output specification, the following functionalities will be implemented for every language:

- The *Template Extractor* will accept standard unprocessed (ASCII or HTML) texts as input. These texts come either from the user interface (query text) or they are delivered by the Document Manager (documents contained in the registered touristic sites). The template extractor returns either a single template or a set of templates, which are either stored in the template DB (if the source text belonged

[3] The definition of the objectives and the methodology of MIETTA was the result of a close cooperation among all partners. Thus what I expose in this section has to be considered a "shared" work. All errors are mine.

to some tourist site) or matched against the templates contained in the DB (if the source text comes from a user query);

- The *Template Matcher* takes as input a query template and, by using the global database of templates as a knowledge source, returns a list of pairs where each pair contains a pointer to some document in the relevant tourist site and a set of templates.
- The *Report Generator* takes input from the template matcher and returns a text to be passed to the user interface;
- The *Relevance Engine* applies advanced indexing methods (e.g. concept-based indexing) to store contents of all the relevant items in the document base, and co-operates with the template handler in order to provide better output to the users.

In fig. 3 the standard interaction with the user is described: the user addresses a Query which is immediately passed to the Query Manager. The Query Manager sends the query to the Template Extractor, which transforms it into a template, which is passed to the Template Matcher. The Template Matcher interrogates the Template DB to retrieve matching templates. Then it passes them back to the Query Manager. The Query Manager evaluates if the answer from the template system was satisfactory or not. If not, it addresses the Query to the low level Relevance Engine which performs a search in the index and send the retrieved documents back to the Query Manager. Once all this processing is done, it sends everything to the Report Generator, which, for every document, generates a short natural language description to be returned to the user through the User Interface.

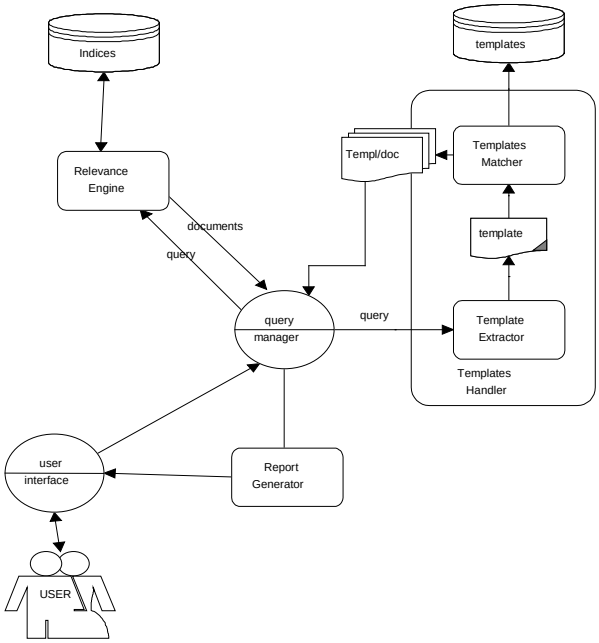

Figure 3: The process of querying a tourist server.

In fig. 4 we describe a typical process of updating the knowledge base of the server. The various regional DBs/web sites notify the Document Manager of the presence of

new documents. These are retrieved by the Document Manager, which passes them to the Template Extractor. The templates that are obtained are stored into the template DB and enrich the knowledge base of the system. In parallel the documents are indexed through advanced technologies of information retrieval for later retrieval, in case the information extraction engine can not provide any satisfactory request to some user's query.

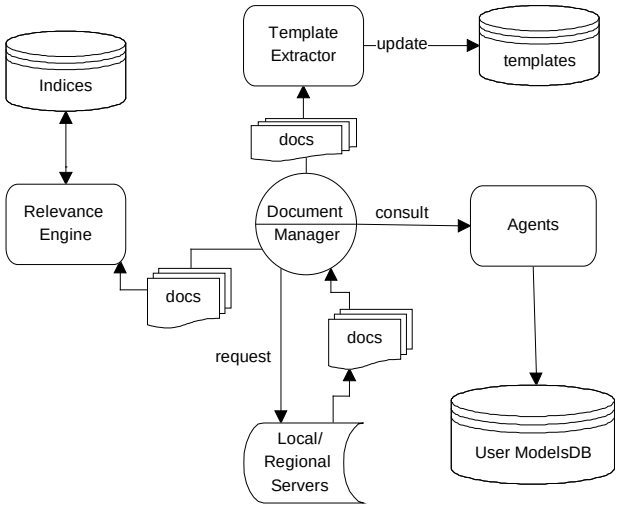

Figure 4: The process of updating the knowledge base of the tourist server

3.2 THE DESIGN OF TEMPLATES

It is evident that the templates used in the MIETTA prototype have to be language independent. This is a core assumption of the project, actually one which the whole system is based on. Its violation would cause both poorer retrieval performances (the template(s) extracted from the query would not match the ones contained in the DB of templates) and worse performance in the document presentation phase (not all templates would undergo generation in the target language as they could still contain information specific to the source language). Consequently, the design must conform a commonly agreed pattern. In order to make this possible, an abstract description language, such as the one of typed feature structure, will be used. This will enable the structuring of templates in a hierarchical way, in order to minimize the amount of information. Moreover it will allow the use of standard unification techniques (coupled with domain dependent inferences) in order to perform template matching for retrieving the templates satisfying the constraints expressed by the user query. Finally, such a structuring allows an easy porting of the set of templates into commercial databases (either relational or object-oriented) for more efficent retrieval in a real life setting. Figure 5 (mainly due to Paul Buitelaar, Klaus Netter, Feiyu Xu) shows an example of a hierarchical structuring of three templates, where principles of feature inheritance are applied in order to keep information compact.

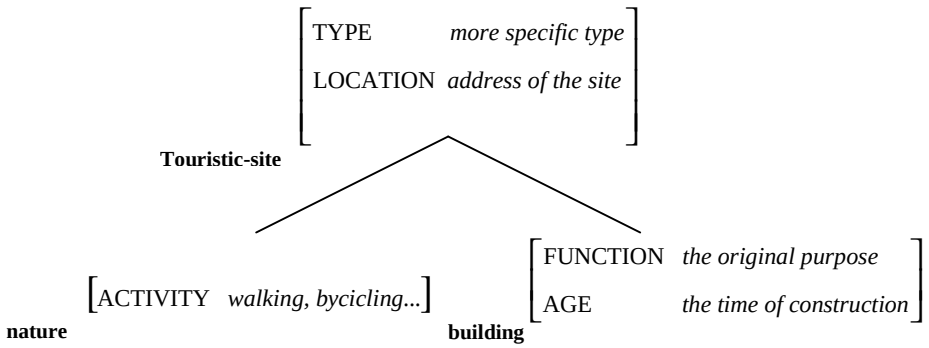

Figure 5: Some sample templates

3.3 INFORMATION EXTRACTION ENGINES

In order to fulfill the goal of the MIETTA prototype, the consortium has rejected the use of deep analysis modules based on full parsing by competence grammars. Indeed, this kind of technology has been considered as not yet mature for the processing of large amounts of real-life texts. Shallow systems, on the contrary, have been proved to be accurate and efficient enough to be successfully applied in information extraction tasks. These methods usually combine well-mastered NLP technologies with statistical methods.

The generic category of shallow systems actually embraces two different kinds of analysis, which are in most cases integrated. On the one hand there is the shallow syntactic analysis, which mostly reduces to the identification of the main constituents of a sentence (VPs, NPs, PPs, etc.). On the other hand, shallow parsing techniques are almost invariably coupled with conceptual, or semantic, modules, which take advantage of the previous shallow representation in order to produce templates representing part of the information encoded in the input text. These modules are usually domain dependent and are implemented either as sets of rules which activate a certain semantic template when a certain syntactic pattern is retrieved, or as finite state automata which apply when a certain semantic trigger is encountered.

The three language technology partners of the MIETTA consortium have agreed to reuse for this particular task three information extraction engines (or parts thereof) which had been already developed for different purposes. In particular CELI will reuse IUTA (Italian Unrestricted Text Analyzer, cf. [2]), DFKI will reuse SMES (cf. [4]) and University of Helsinki will reuse a combination of already available NLP tools (cf. [5]).

SMES is an information extraction core system for real world German text processing. The basic design criterion of the system is of providing a set of basic, powerful, robust and efficient natural language components and generic linguistic knowledge sources, which can be easily customized for processing different tasks in a flexible manner. SMES consists of the morphological component MORPHIX, a declarative tool for expressing finite-state grammars, and an efficient and robust bi-

directional lexically driven parser. The information extraction tool for Finnish is based on three existing modules: (1) FINTWOL: a morphological analyzer of Finnish, based on Kimmo Koskenniemi's language-independent two-level morphology; (2) FINCG: Fred Karlsson's Constraint Grammar parser for Finnish, which performs a morphological disambiguation of ambiguous word-form tokens and gives them a surface-syntactic analysis; and (3) a low-level syntactic Constraint Grammar tagger for Finnish which identifies the noun phrases.

As far as IUTA is concerned, the system will be described in section 4 of this chapter.

3.4 NATURAL LANGUAGE GENERATION

Hovy ([6]) distinguishes three types of generation task: text planning, sentence planning, and surface realization. "Text planners select what information to include in the output from a knowledge pool, and out of this create a text structure to ensure coherence. On a more local scale, sentence planners organize the content of each sentence, massaging and ordering its parts. Surface realisers convert sentence-sized chunks of representation into grammatically correct sentences." The processes used in NL generation are classified and ranked according to sophistication and expressive power, starting with inflexible canned methods and ending with maximally flexible feature combination methods.

In the context of the MIETTA system the template generator JTG/2 from CELI/DFKI will be adopted by the consortium. JTG/2 is a porting into Java™ of TG/2 (developed in LISP at DFKI [7]), a surface generator that can be easily hooked up to application systems or to language planning systems. In domains where no extensive utterance planning is needed, TG/2 can be parametrised according to preferred properties of utterances (style, wording, and contents). Parameters guide the selection of grammar rules, which are condition/action pairs allowing the grammar-writer to formulate canned text, templates, or context-free derivations, as well as any combination thereof.

The major challenge for the adopted NL generation engine will be represented by the language independent nature of the templates. Indeed, the set of templates retrieved after a query matching will be passed directly to the generator. Two main tasks can then be envisaged:

Presentation: the generator will have to arrange all the information in order to present the user with a reasonably structured page. In order to accomplish this goal, the generator will take advantage of the fact that the generated pages will be in HTML. In this way, some of the relations which would have to be expressed through natural language (e.g. enumeration, specification, topic proximity) can be encoded using typical hypertextual/graphic devices.

Sentence generation: since language independent templates are passed to the generation module, the task of the generator is particularly difficult, as it has to reconstruct every single piece of linguistic information. Moreover, given the "richness" of the input templates, it will have to keep a correspondence table mapping abstract concepts to language dependent lexemes. These lexemes will be used to perform lexical insertion after the phase of sentence plan.

4. UTA and IUTA

UTA (Unrestricted Text Analyzer) is a general purpose Information Extraction engine developed at CELI. It is based on an object-oriented architecture in which language independent modules satisfying certain predefined interfaces can be plugged without any need to access their internal code (cf. [8]). In this sense it would be more reasonable to talk about an architectural model than a program in the proper sense of the word: indeed the design of every component and the interfaces they have to satisfy force a strictly modular view, which enables a fast prototyping of ready-to-deliver information extraction systems. [4]

UTA comes together with a set of *engines*, which includes a textual tokenizer, a class based morphology, a part of speech tagger, a shallow parser system (chunking), a system for extracting underspecified logical forms, and a module for coding simple domain dependent rules for template filling. In the design of the modules we stuck to the assumption that no language dependent information should be contained in the processing engines. So, they are in principle extensible to deal with any language for which there are enough linguistic resources.

UTA has been used to produce IUTA, an Italian information extraction system (cf. [2], [9]).[5] IUTA exploits all the modules currently available to UTA thus implementing a quite prototypical processing flow, as described in figure 6.

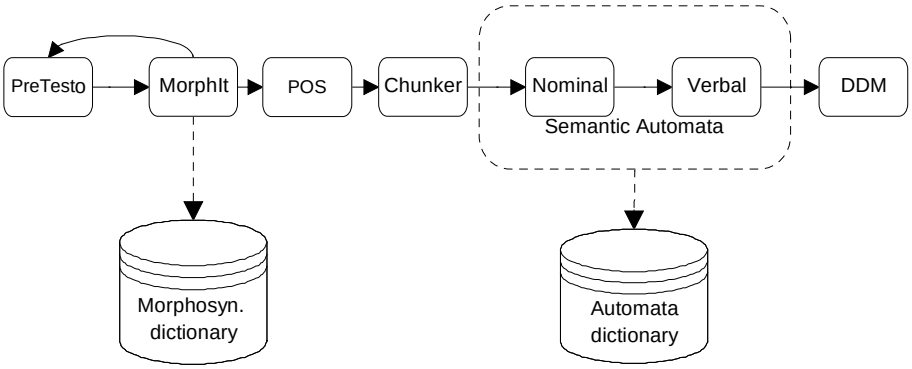

Figure 6: The architecture of IUTA.

[4] This result is achieved by designing every module as a JavaBean™ component and by establishing a common communication protocol. In the optimal case a whole system can be parametrized just with the help of a visual tool for JavaBean manipulation.

[5] Actually a version of IUTA was already existing when UTA was under development. The lingware of the previous version was just migrated in the new architectural model.

The first modules (Pretesto, Morphit, the POS tagger and the chunker) are quite standard, so they do not deserve a particular attention (cf. [10]). In the next section we will rather describe in some detail the core of the IE system, i.e. the set of semantic automata and the Domain Dependent Module).

4.1 THE IE CORE OF IUTA

The basic idea of IUTA is that effective and easily parametrizable IE engines should be based on two processing steps. During the first step a sequence of chunks is analyzed in order to produce a *Naif Logical Form* (*nlf*), i.e. an abstract representation level independent on the particular type of application for which the system has been designed. Information extraction in a proper sense is performed only during the second step of processing, where *nlf* are spanned by template driven rules which select the appropriate template and fill the relevant roles. The main advantage of this strategy over approaches which goes directly from syntactic structures to filled templates (cf. [11], [12], [13], [14]) is reusability: indeed part of the processing needed to bridge from syntax to templates is kept common to many applications, in such a way that linguistically motivated improvements are immediately reflected on all the system accessing the intermediate level of *nlf*s. The advantages from the point of view of software engineering are significative, as it is possible to maintain a certain degree of coherence among different application (cf. fig. 7)

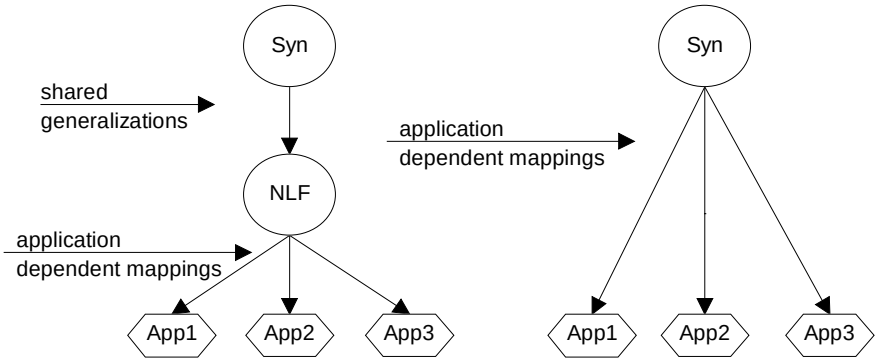

Figure 7: Advantages of NLF.

More technically, the module in charge of producing *nlf*s is implemented as a cascade of finite state automata (FSA) having as an alphabet indexed objects. The indices associated to every object can be referred to in a separate part of the FSA, called *action*, whose main task consists in building an appropriate output once the associated automaton reaches a final state. Consider for instance the following sample rule:

```
(PP$1|ADVP$1)* NP$2 VT$3 (PP$1|ADVP$1)*
Action = {TransNLF($1,$2,$3)}
```

The regular expression described in the first line is compiled into a deterministic FSA[6] to be applied to the input sequence, represented in this case by a sequence of objects of type `Chunk`. If the application has success, the objects referred to by the variables prefixed with "`$`" (`$1` an array of chunks having category PP or ADVP, `$2` a chunk with category NP and `$3` a chunk with category VT)[7] are passed directly to the Java constructor `TransNLF` which takes care of producing the proper output.

In the actual setting we use only two levels of FSA, one for guessing the semantics of nominal expression and one for identifying the *nlf* to be associated with verbs. However more levels can be easily implemented, for instance for describing parenthetical expressions and subordinate clauses.

As for the modules for template filling, labeled *Domain Dependent Module* (DDM) in fig. 7, it is conceived as a set of template-driven structure matching rules, along the lines described in [15]. Basically, for each attribute of each template *type* there is a set of rules with the task of spanning the input structure (a *nlf*) in order to find a possible filler for that attribute. The rules are ordered in terms of preference and the first rule which fires determines the values that the attribute will have in the resulting template. For instance the following rule spans a *nlf*, looking for the subject of a verb of a certain kind and sets the role **company** accordingly.

```
[PRED ricercare,
ARG1
    [ PRED $1]
]
action = {theCurrentTemplate.setCompany($1)}
```

DDM is also the module where textual relations and anaphoric links are solved, depending on the requirements of the application. Rules for performing these tasks are encoded directly as Java methods taking as input the filled templates.

5. Conclusions

We have described a model architecture for a system of information extraction, showing how it is exploited in order to satisfy application needs emerging from a field such as tourism. The system is based on the interaction of a set of monolingual IE systems with a set of natural language generation engines. In order for the system to be truly multilingual we have shown that a big emphasis has to be put on the design of templates rather than on the processing chain itself. Indeed a project such as MIETTA proves that different IE system can cooperate to reach the goal of multilingual access to monolingual information just by virtue of an agreement on appropriate communication standards and on a strict design of templates.

---

[6] In order to increase the efficency of the system all the rules belonging to the same level are compiled together to form a unique deterministic FSA.

[7] Actually the expressive power of our rules goes beyond this, as it is possible to refer to any feature of the object, provided the set of the possible values of such a feature is finite.

References

[1] Salton, G. Automatic Processing of Foreign Language Documents. *Journal of the American Society for Information Sciences*, 21:187-194, 1970.

[2] Bolioli A., Dini, L., Di Tomaso, V., Goy A., Sestero D., "From IR to IE through GL", in *Proc. of Int. Workshop on Lexically-Driven IE*, Frascati, Italy, 1997.

[3] McKeown, K.R., *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Studies in Natural Language Processing. Cambridge University Press. 1985

[4] Neumann, G., Backofen, R., Baur, J., Becker, M., Braun, C., An Information Extraction Core System for Real World German Text Processing, in *Proceedings of ANLP 1997*.

[5] Koskenniemi, K. Representations and Finite State Components in Natural Language, In Roche, E. and Schabes, Y. (Eds) *Finite-State Language Processing*, pp 383-402., The MIT Press, 1997.

[6] Hovy E.,. *Natural Language Generation*. In: Varile G., Zampolli,A. (Managing Editors), *Survey of the State of the Art in Human Language Technology*, 1995.

[7] Busemann S., Best-first surface realization. In Donia Scott (Ed.), *Eighth International Language Generation Workshop. Proceedings*. 1996.

[8] Bolioli A., Dini L., Di Tomaso V. UTA: a light portable component architecture for IE, forthcoming.

[9] Bolioli A., Dini L., Di Tomaso V., Goy A. and Sestero D. MILK: a Hybrid System for Multilingual Indexing and Information Extraction. *Proccedings of Recent Advaces in Natural Language Processing* (*RANLP 1997*), Tzigov Chark, Bulgaria, 11-13 September 1997.

[10] Bolioli A., Dini L., Di Tomaso V., Goy A. and Mazzini M., IUTA: un sistema di estrazione dell'informazione per l'italiano. In *Apprendimento automatico e linguaggio naturale*. AI*IA. 1997.

[11] Hobbs J.R., Appelt D.E., Bear J., Tyson M., "Robust Processing of Real-World Natural-Language Texts", in *Proc.of 3rd Conference on Applied NLP*, Trento, Italy, March, 1992.

[12] Lehnert W., Cardie C., Fisher D., Riloff E., Williams R., "University of Massachussets: Description of the CIRCUS System as Used for MUC-3", *Proc. of 3rd MUC*, San Diego, CA, 1991.

[13] Appel D., Hobbs J., Bear J., Israel D., Tyson M., "FASTUS: A Finite State Processor for Information Extraction from Real World Text", in: *Proc. of IJCAI*, 1993.

[14] Hobbs J.R., Appelt D., Bear J., Israel D., Kameyama M., Stickel M., Tyson M., FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. In Roche, E. and Schabes, Y. (Eds) *Finite-State Language Processing*, pp 383-402., The MIT Press, 1997.

[15] Dini L., CIRO User Manual, *DFKI Internal Report*. 1997