FACILE: DESCRIPTION OF THE NE SYSTEM USED FOR MUC-7

William J Black, Fabio Rinaldi and David Mowatt
Department of Language Engineering
UMIST
PO Box 88, Sackville Street
Manchester M60 1QD, United Kingdom
bill@ccl.umist.ac.uk

INTRODUCTION

In this paper, we describe the system used by the UMIST team as members of the FACILE consortium, to undertake the NE task in MUC-7. The main characteristics of this system employed are as follows:

- it is rule-based
- its rule formalism supports context-sensitive partial parsing
- rules may use pattern-matching-style iteration operators
- the notation is much more readable than classic pattern-matching languages
- rules can be assigned an explicit weight which is used in choosing between competing analyses.
- there is a method for identifying name-strings as coreferential with longer variants in the same text
- the system does not employ learning techniques

The development of the system began only about 20 months ago, so it has not been used in any previous comparable trials. We looked forward to slightly higher scores than we obtained in the formal run, because at the dry run stage, we had obtained almost identical scores to our best results with training data.

In the rest of this paper, we first give some background on the context in which the system used in the MUC-7 NE task was developed. We then outline its internal structure, concentrating on the rule notation which is its most salient feature. An evaluation of its performance in the task then follows, before concluding with some speculation on the extent to which the approach adopted is susceptible to further improvement.

BACKGROUND

UMIST's participation in the MUC-7 NE task was conducted with a recently constructed module of a larger system whose main purpose is text categorization. The FACILE project, (Black et al 1997) co-funded by the European Community's Language Engineering programme, is a precompetitive industry-academic collaborative project. Its main task is the filtering of news by fine-grained knowledge-based categorization. A version of the FACILE system has been deployed for several months in a news filtering service offered by an Italian News Agency, Radiocor.

It was clear from earlier experience in the COBALT project (Gilardoni et al, 1995) that other modules would benefit from a component which could identify the often complex proper names which occur frequently in financial news texts. Taking an externally sourced name-finder for English was not an ideal solution because the FACILE system categorizes texts in four languages: English, German, Italian and Spanish. We required name-finding to be done in all four languages, using a standard interface with a morphological analyser and tagger for those languages. A rule-based

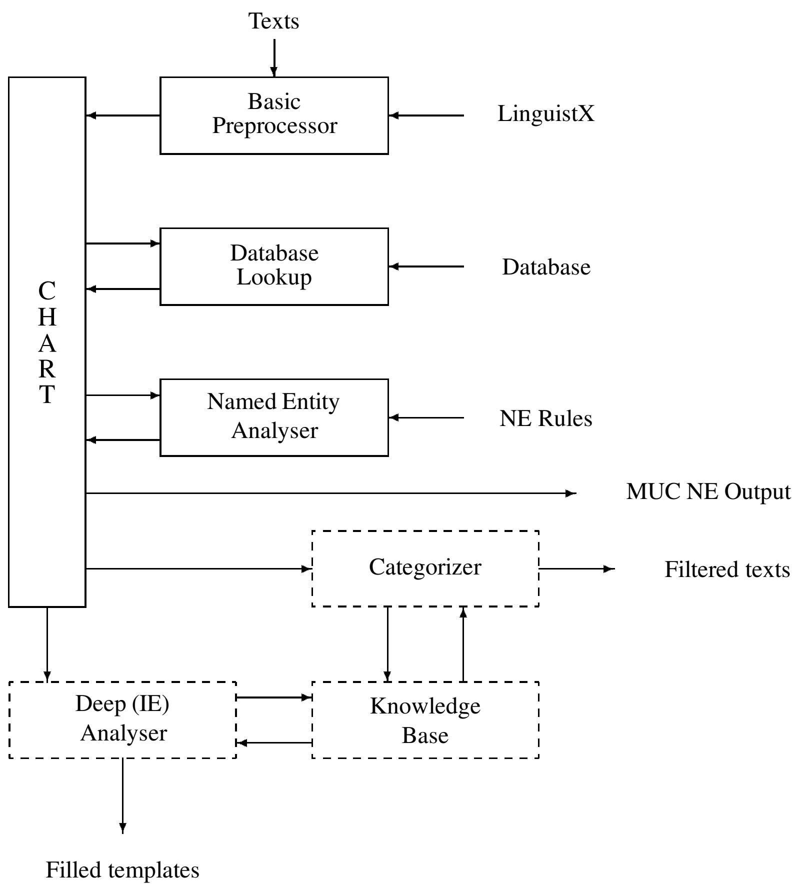

Figure 1: System Architecture

component was constructed which reflected the approach of Coates-Stevens (1992), but within the software framework adopted for FACILE. It is this component which has been tested against the MUC-7 NE task.[1]

In addition to the categorization task, the FACILE system's functionality includes information extraction as understood in the MUC context, using a linguistically-motivated system developed by our partners, (Ciravegna, 1995; Ciravegna, Lavelli and Satta, 1997) and which has been very successfully applied to the analysis of Italian texts.

At UMIST we had to put our effort into completing and indeed revising the NE analyser, at the expense so far of the English resources for the information extraction component. This has meant that it has not been possible to participate in MUC-7 to the extent originally envisaged, since the IE component's adaptation to English is not yet available.

SYSTEM ARCHITECTURE

The FACILE preprocessor accepts input to the system, normalizes the text, recognizes special formatting, tokenizers, tags, looks up single and multi-word tokens in a database, and carries out proper name recognition and classification. The output forms the input to other FACILE modules (the Shallow Analyzer and the Deep Analyzer, neither of which was used in the MUC-7 NE task).

Basic Preprocessor and Database Lookup

[1] The same component was subjected to a highly experimental trial in the TE task, in which the results obtained reflect the NE analyser's ability unaided to extract no more than the name and category of some of the entities mentioned in the text.

The preprocessor utilizes the InXight LinguistiX tools as a third-party component. Through a functional interface, the preprocessor is able to utilize these tools for tagging and morphological analysis. The proven finite-state technology that the tools employ ensures the necessary speed, reliability, coverage and portability. Modules developed within the project carry out text zoning, tokenisation, database lookup and named entity rule application.

FACILE treats tokens as feature vectors. The follow-up modules derive all information about a token exclusively from its corresponding feature vector.

The feature vector stores the following information about a token: where it begins and ends as character offsets, what separates it from its predecessor (white space, hyphen etc.), what text zone it comes from, its orthographic pattern (capitalised, all capitalised, mixed, lower case etc.), the token and its normalised forms, its syntax (category and features), semantic class (as obtained either from the database or morphological analyser), morphological analyses, partitioned into those consistent with the tagger's choice and others (for possible use by other modules). (1) is an example, in LISP notation.

(1)
```
(1192 1196 10 T C "Mrs." "mrs." (PROP TITLE) (^PER_CIV_F)
(("Mrs." "Title" "Abbr")) NIL)
```

In this example, the separator is octal 10, the text comes from the main body, the token is capitalised, literally "Mrs.", normalised to "mrs.", syntactically a PROP and TITLE, and semantically according to the database a prefix (^) for a female civilian person. On the second line are the results of the morphological analysis.

The preprocessor fills out the feature vector from various sources. The first six fields shown above are obtained from the text using the text zoner and tokenisation modules. The normalised form field comes either from the morphological analysis (see below) or from algorithmic procedures for the handling of numeric tokens. The syntax field comes from the morphosyntactic tagger, and the morphological analysis from the morphological analyser. The latter typically offers several alternative analyses and the full list of results is partitioned into those that are consistent with the tagger's decision and those that are not. The semantics field comes from lookup in a database which has information on words and phrases belonging to the categories of named entities themselves as well as to categories that occur as prefixes or suffixes of names. Where there exist database entries for multi-word tokens, the single words are replaced by a single token vector for the compound.

The structure of this table differs from the data structure used in chart parsing in that there is only one initial edge per token. Alternative analyses are packed into the SEM and other-morph fields.

RULE-BASED NAMED ENTITY RECOGNITION

As mentioned earlier, we had found the approach to named entity recognition described in Coates-Stevens (1992) interesting, but that system was coded in Prolog, which was not one of the agreed languages for implementation in the FACILE project.

Our first thoughts were to use a pattern-matching language like FLEX or PERL, starting from the tagger output as word/tag pairs. However, first attempts to use PERL led to unreadable patterns of many full lines in extent because of the number of features to take account of. The need to handle coreferences and scores, and the slowness of PERL pointed to a more complex interpreter, and so we came to specify a more congenial notation. Because in any one pattern constituent we would need to refer to only one or two of the properties, we chose to use a attribute-value notation for readability, but not one as powerful as we might want to use for a full syntactic and semantic analysis.

The rule notation

Two versions of the rule language and its interpreter have been defined, the current version having been proposed and implemented following an evaluation of our performance in the dry run. This description refers only to the current version.

Rules have the general form $A \Rightarrow B\backslash C/D$. A is a set of *attribute operator value* expressions, where the values are atomic expressions, disjunctions (using the operator $|$) or negated atomic expressions or disjunctions, i.e. a *one-level* attribute-value matrix (AVM). In an individual attribute operator value expression, the left hand side may be any of ten specified columns in the input token vector, or an additional attribute if the matching token has been found by rule. B, C and D are sequences of such one-level AVMs (B and D possibly empty, since they constitute the left and right context of C). Each AVM is optionally followed by an iteration specification, *,+,?, or $\{\langle \textit{integer} \rangle, \langle \textit{integer} \rangle\}$. AVMs may be grouped by parentheses to allow for the iteration of a sequence of constituents, although in the current implementation, recursive grouping by parentheses is not permitted.

The left-hand side of a rule may also have a score in the range -1 ...+1, which defaults to 1. # is the comment character.

The comparison operators include =, != (not equal), <, <=, > and >=. If the value in the chart edge is a disjunction (i.e. list) the = operator is satisfied if any of the members of the list is identical to the value in the expression (or any one of the values if that is a disjunction). The negation operator != is satisfied if there is a null intersection between the values in the edge and those in the expression. Substring comparisons are permitted by the inclusion of wildcards in the value expression.

Any variable (a symbol whose print-name begins with "_") which occurs on the right-hand side of an expression is unified with all other occurrences of the same variable in the rule. This can be used to transfer specific information to the left-hand side as well as to enforce constraints.

An example of a rule is (2), which is satisfied by a token whose normalised form (i.e. modulo capitalization) is "university," the literal "of" and a location or city name. A string matching this description is assigned the syntactic tag PN and the semantic tag ORG.

(2)
```
[syn=NP, sem=ORG] (0.9) =>
 \ [norm="university"],
   [token="of"],
   [sem=REGION|COUNTRY|CITY] / ;
```

Example (3) shows how the right context may be suggestive of the tag to assign, with a relatively low certainty factor to take account of this. The target pattern is a single upper-case token which the tagger guesses to be a PN, and not a PN that has been found by applying a rule. In this example, the right context is expressed literally instead of using SEM or SYN values. If the whole pattern is matched, the arbitrary additional attribute `orgn` receives the value of the `token` in the first constituent through unification of the instances of the variable *_O*.

(3)
```
[syn=NP,sem=ORG, orgn=_O] (0.5) =>
 \ [syn=NP, orth=A, token=_O, source!=rule]  /
   [token="said"|"announced"],
   [token="it"],
   [token!="was"|"is"];
```

Coreferent names

The purpose of the variable in rule (3) may not be immediately apparent. Variables were first introduced because of the need to treat coreferences between instances of the same name. Whilst an extended form of a name may be used on its first mention in a text, it is typically not used again in the same text. The referent of the phrase "Foreign Secretary Robin Cook" will normally be mentioned subsequently as "Mr Cook," for example. "Mr Robin" would not be possible (except in the households of aristocrats or in an old family firm). The variable allows the repeatable part of the name to be identified for matching with any subsequent mentions, as made clear by Rule (4).

(4)
```
[syn=NP,sem=PER,title=_T,surname=_S] =>
   [sem=^PER,token=_T],
 \ [orth=C,sem=FIRSTN],
   [orth=A|C]?,
   [orth=C,token=_S] /;
```

Rule (5) illustrates the coreference operator >>, which stipulates that there be an antecedent constituent matching the AVM following the operator, which can satisfy the variable bindings established in the rest of the rule. In this case, both surname and title must be identical with their antecedents.

(5)
```
[syn=NP,sem=PER] (0.7) =>
   [sem=^PER,token=_T]
 \ [orth=C,token=_S] /
   [orth!=C]
 >>
   [sem=PER,surname =_S,title=_T];
```

When a coreference of this type is found, in addition to the explicit assignment of the `syn` and `sem` fields and the `title` and `surname` attributes, an `ante` field of the newly found constituent is assigned the unique identifier of the matching antecedent. By using variables in this way, we could in principle deal with coreference relationships that are made explicit syntactically, such as that between a name and a description in apposition. However, we have only explored this possibility to a limited extent in our experimental application of the system to the TE task.

Comparison with other pattern-matching languages

Standard pattern-matching languages like PERL, FLEX, SNOBOL etc., are designed to process surface patterns in text. Where tagged text is to be pattern-matched, it is possible to pair tags with words as in `the/AT cat/NN sat/VBD` etc. and to define patterns over these sequences. However, as we have pointed out, more than just the literal token and its syntactic tag are relevant to the NE recognition problem. To make all of these properties into facets of a token structure makes the statement of the rules in "raw" PERL hopelessly long-winded and error prone. One important effort at producing a higher level language that PERL is "Mother of PERL" (MOP) - see Doran et al (1997). It enables the pattern writer to focus on a single or a few attributes at a time, but does this by the use of several separate layers of rules. Our language by contrast, allows conditions to refer to attributes arising from multiple levels of analysis in a single rule-set. However, in our view, a more significant advantage of our own rule language is its readability and accessibility to the rule-writer. We have in comparison far fewer symbolic operators and instead use a variant attribute-value matrix notation to make the individual rules more self-documenting.

The rule interpreter

The current implementation of the rule interpreter is adapted from a left-corner chart parser, although we have considered compiling to finite state machinery if it proved too inefficient.

Although the basic rule-invocation strategy is bottom-up, partial parsing in this way could lead to runaway recursion with some rules with a single constituent (except for left and right context). This can arise when a rule has the pattern indicated in (6), or wherever the rhs is underspecified enough to accept a constituent with sem=CAT.

(6)
```
[...sem=CAT...] =>
  ...\ [...sem=CAT...] / ...;
```

For this reason, the rule interpreter is depth-limited.

With the right-hand side of rules including context as well as the phrase to be matched, rule-invocation is more complex than in the left-corner algorithm, since the scanner can have moved on by the time a needed inactive edge is added.

Data Structures

The working data structures of the interpreter include an active chart, comprising sets of active and inactive edges and a vertex index. In addition, there is a property-value table which stores the values of any attributes not in the standard

Table 1: Simplified chart extract

Edgeno	Token	Sem
1	Mr	ˆPER
2	John	FIRSTN
3	Smith	
4	John Smith	PER

Table 2: Property-value table

Property	Value	Edgeno
surname	Smith	4
title	Mr	4

chart columns. This table is a simple hash table, and can be illustrated as follows: Table 1 shows a simplified chart with initial edges 1–3 and added (inactive) edge 4 produced by application of Rule 4. The values of the `title` and `surname` fields are stored in the property-value table Table 2. This is indexed both by edge number and property, avoiding the need to search for antecedents.

The main algorithm

The active chart mechanism has been extended to deal with iterable and optional constituents. An iterable or optional constituent that can match the current token gives rise to two new edges, one an active one in which there can be more iterations, and another in which the cursor advances to the next constituent or concludes the rule. To make this process more efficient, active edges do not contain copies of the right hand side of the rule, but merely pointers to rules, a state vector referencing the current constituent group and constituent within the group, and bindings for any variables in the rule.

Advanced rule-invocation strategy

A working set of NE recognition rules may easily be over one hundred in number. If every rule within a rule set were to be tested against every edge of a document then the document would both take far longer to process than if some form of selection algorithm takes place. This is the role of the advanced rule invocation strategy - computing, for each edge of the document, which rules will definitely not fire and which rules have a chance of firing.

The algorithm works by assigning properties to both rules and document edges when the rules and document are read in. Consider rule (7) which cannot be completed if there are no cardinal numbers in the edges it will subsume. Similarly, Rule (8) cannot be completed if there are no ’interesting’ properties in the semantic field of the next few edges.

(7)
```
# 20.5 million
[syn=NN, sem=NMB, norm="(* _C _N)"]  (0.8) =>
 \ [syn=CD, norm=_C],
   [token="million"|"hundred", norm=_N] / ;
```

(8)
```
# Washington officials
[syn=NP, sem=LOC, place=_P] (0.8) =>
 \ [sem=LOC|CITY|AIRPORT, token=_P] /
   [norm="official"];
```

When a rule is read in, it is assigned a ’requirement’ value which indicates whether, in the next 3 edges, the rule will require (a) A cardinal number (b) A capitalised or an all capitals token, or (c) An ’interesting’ semantic property - i.e. ORG, LOCˆ, PER_CIV, DATEUNIT but not NULL, SGML, PUNCT etc.

When the edges of a document are read in, they, similary, are assigned a 'property' value according to the properties of the edge. If an edge contains a capitalised word and is tagged as an organisation (sem=ORG) then then 'property' value will indicate this. Both 'requirement' and 'property' values are stored in binary arrays.

When the main loop is initiated, the properties of the current edge and the following two edges are added together. Before any rule is fired, this 'property' value is checked against the requirements of each rule to make sure that the rule has at least a chance of completing.

In the tests we have done, this preselection of which rules fire reduces the total run time by approximately one half, with no loss of accuracy.

The preference mechanism

As noted above, rules have a default certainty of 1, or an assigned certainty in the range -1 to 1. If rules give competing descriptions for the same span of the text, the `sem` value with the highest score is preferred. Where several rules come to the same conclusion about the `sem` value, evidence combination comes into play. We combine such scores using Shortliffe and Buchanan's (1975) Certainty Theory formula. C_{bf} represents the initial certainty of a proposition or that based on accumulated evidence so far. C_X is the certainty value for the same proposition, attributable on the basis of a new rule X, not previously considered. C_{cf} represents the cumulative certainty after assimilating C_X and C_{bf}. (9) shows how the formula applies in combining two positive certainties. For reasons of space, we omit the other cases here.

(9) $$C_{cf} = C_X + (C_{bf} * (1.0 - C_X))$$

After evaluation of certainties for each given text span, a further preference is applied which prefers longer spans to shorter in cases of overlap.

The resulting analysis is a single semantic and property description of each identified name expression in the text. For our integrated system's categorization and template filling components, these are interspersed with the other expressions in the text with their tags and morphological analyses. For the purposes of MUC evaluation, reports are generated in the appropriate format.

WALKTHROUGH

NE Task

The walkthrough document is representative of our overall performance in the formal run.

There are 11 missing entities. "MURDOCH" is missing three times although we capture it later in the main text. The three missing occurences are all the beginning of the article. We didn't have this surname in the database, so the only way to have identified this as a person name would be coreference with the instances in the body text.

Unfortunately the current implementation finds only backward references. That is adequate in the main text since normally a new name is "explained" by a descriptive phrase the first time it is mentioned. However it is possible to have mentions of the name in the title and the preamble before any explanation is given. The way we intended to cope with this problem (following the example of some of the MUC-6 systems) was to process SLUG and PREAMBLE after the main text. However in the version of the NE analyzer that we used for the final run this approach was not properly implemented.

We miss the four occurrences of "Grupo Televisa" and "Globo". This can be explained by the fact that we don't have in our DB the suffix "SA" as a company designator.

We also have some heuristic rules which can pick up partial descriptions in apposition to a name, but here again the crucial clue-words and phrases 'broadcaster", "publisher" and "media conglomerate" were not in our database. There are certainly sufficient clues of this nature in sentence (10).

(10) “Grupo Televisa SA, the Mexican broadcaster and publisher, and the giant Brazilian media conglomerate Globo”

(11) “Llennel Evangelista, a spokesman for Intelsat”

(12) PERSON, a spokesman for ORGANIZATION

We miss another person name which could have been easily captured. In sentence (11), it is possible to identify the pattern (12). This would require two separate rules. While we have the rule that captures the organization (13) we simply forgot to insert the corresponding rule to capture the person.

(13)
```
[syn = PROP, sem = ORG, zone = _X] (0.8) =>
  [token = "a", zone = _X] ,
  [token = "spoke*"] ,
  [token = "for"]
 \ [orth = C, norm = _O],
   [orth = C]* /
   [zone = _X] ;
```

We miss the location “Xichang” which again could have been identified in “Xichang launch site” if domain-specific clue words (“launch site”) had been inserted in the DB.

As for the other two missing entries “Hughes Electronics” and “within six months” the first is explained by the missing clue-word “electronics” and the second by the fact that we did not consider “within” as an identifier in rules for Time expressions.

While we get all the occurrences of “New York Times News Service” we miss all the occurrences of “N.Y. Times News Service”. This is easily explained by an error in the rule that identifies it (see example 14). Possible values (among others) for the “orth” flag are C (first capital letter) and A (all capital letters). In the specific case of “N.Y.” it assumes the value O (other, because it contains letters and dots). Simply altering the value of the feature to “C—A—O” would solve the problem. This error also causes the spurious occurrence of “N.Y.” as a single location.

(14)
```
[Syn = PROP, sem = ORG, zone = _X] (0.9) =>
\ [orth = C|A, sem = LOC, zone = _X]+,
  [sem = ORG],
  [norm = "news"] ?,
  [norm = "service", zone = _X] / ;
```

Two occurrences of “March” in “Long March” (referring to a chinese rocket) appear incorrectly tagged as a date. While it is corret that they should be initially tagged as a date, we had inserted in the system rules that capture “artifacts” like this one thus superseeding the initial semantic value. In the specific case that did not work, probably for some yet undetected error.

In “2 p.m. EST” we identify correctly only “2 p.m.” (the suffix EST had not been inserted in the DB).

There is an instance of CNN that we (correctly?) tag as an organization but it is not tagged as such in the keys (an annotator error ?).

“Hughes” is twice tagged as a LOCATION istead of an ORGANIZATION. We haven’t yet tracked this one down.

In “Time Warner” we managed to identify only “Time” [difficult to say how we could have done it without having the whole expression in the DB] and in ”LaRae Marsik” we identified only ”Marsik” (probably becayse we don’t have in the DB “LaRae” as a first name).

The following two organizations had been tagged as persons:

”Home Box Office” ”Turner Broadcasting System”

They appear in the sentence “Time Warner’s Home Box Office and Turner Broadcasting System were among the companies that had leased space” and are picked up by a low-score rule meant to capture conjunctions of people names.

Finally there are two occurrences of “Tele-Communications”, tagged as persons and this is caused again by a low-score rule that considers “said Tele-Communications” in the sentence (15) as evidence for classifying it as a person, as in ‘ “Right!” said Fred.’

(15) “Ms. Marsik said Tele-Communications and its partners”

TE task

We will not discuss in detail the results of the TE task for the walkthrough article because as noted in footnote 1, the scores indicate little more than the contribution made by NE analysis, i.e. finding strings and their categories.

In respect of locations, the value of recall for the slot COUNTRY is particularly low because we relied on quite a small table in lieu of a full-scale “Geographical DB” of Towns and Regions in order to find the country to which they belong. Furthermore, for the entities of type COUNTRY the LOCALE should have been used as the value for COUNTRY slot. That would have significantly increased the recall for this slot (but only a couple of percentage points overall).

As for entities, our Recall for descriptors is extremely low because we didn’t have the Information Extraction Module available, with its extensive linguistic coverage. The Entity Names and Categories are also affected by this problem, although we still managed to capture 35 and 15 per cent of them respectively, using assignments to properties via variables. We attempted to adapt the approach taken in the NE task, for instance having a more refined set of semantic tags (PER_CIV/PER_MIL rather than simply PER) in our rules. However we could not perform this “porting” entirely because of lack of time so in many case our responses were as generic as in the NE task. For instance we have “China Great Wall Industry Corp.” classified simply as an ORGANIZATION and not as and ORG_CO.

ANALYSIS AND CONCLUSIONS

Our best efforts at the NE task on the training data achieved 92% recall and 93% precision just prior to the formal run, and with the same data and rules.

Given that in the dry run, we had equalled our previous best performance with the training data, we were disappointed with a fall-off of 6 percentage points in precision and 14 points in recall.

In general the category where we perfom worst is “Organizations” and this can be partly explained by too many domain-dependent rules, and an inadequate database of company designators and clue words.

Judging from the results in the Walkthrough text, almost all of our errors and omissions are easily traceable to either a lack of entries in the database or a lack of rules or conditions in rules. With the software still under development, we probably dedicated no more than a person-month to resource development and testing, and will be able to make considerable further improvements in the coming months. We feel on the whole that the approach is vindicated. We also look forward to being able to use the FACILE IE component to carry our proper tests on all the MUC-7 data in the near future.

Acknowledgments

FACILE (Fast and Accurate Categorisation of Information by Language Engineering) is a collaborative project co-funded by the European Union under contract LE2440.

The participating sites are Quinary SpA (Italy), SEMA Group S.A.E. (Spain), IRST (Italy), Smart Information Services GmbH (Germany), UMIST (UH), ÖFAI (Austria), Italrating (Italy) and Caja de Segovia (Spain).

Various individuals at the mentioned sites have participated in the development of the FACILE system, among them: Luca Gilardoni, Paola Prunotto, Silvia Mazza, Fabio Ciravegna, Alberto Lavelli, Nadia Mana, Jose-Luis Fernandez-Lopez, Maria Jesus Herrero, Kai Schirmer, Johannes Matiasek, Harald Trost.

REFERENCES

[1] W.J. Black, L. Gilardoni, R. Dressel, F. Rinaldi. Integrated text categorisation and information extraction using pattern matching and linguistic processing. *Proceedings of the 5TH RIAO CONFERENCE, McGill University, Montreal, Canada, 25th-27th June 1997*

[2] Ciravegna, F. Understanding Messages in a Diagnostic Domain, Information Processing and Management, Vol. 31, No. 5, pp 687-701, 1995.

[3] Fabio Ciravegna, Alberto Lavelli and Giorgio Satta Efficient Full Parsing for Information Extraction in *Atti dell'Incontro dei Gruppi di lavoro dell'Associazione Italian per l'Intelligenza Artificiale (AI*IA) di Apprendimento Automatico e Linguaggio Naturale (AA-NL '97)*, Torino, December, 1997

[4] Coates-Stevens, S. (1992) The Analysis and Acquisition of Proper Names for Robust Text Understanding. *PhD Thesis*, City University, London.

[5] Doran, C., Niv, M., Baldwin, B., Reynar, J.C. and Srinavas, B. (1997) Mother of PERL: A Multi-tier Pattern Description Language. *Proceedings of the International Workshop on Lexically Driven Information Extraction*, Frascati, Italy, July 16, 1997, 13–22.

[6] Gilardoni, L., Prunotto, P., Rocca, G., Deotto, F. Di Cresce, A.; A news Categorisation system for Traders and Analysts, Third International Conference on Artificial Intelligence Applications on Wall Street, New York, June 1995.