PatEdit: An Information Extraction Pattern Editor for Fast System Customization

Dimitra Farmakiotou, Vangelis Karkaletsis, Ioannis Koutsias, George Petasis and Constantine D. Spyropoulos

Software and Knowledge Engineering Laboratory,
Institute of Informatics and Telecommunications,
National Centre for Scientific Research (N.C.S.R.) “Demokritos”,
P.O. BOX 60228, Aghia Paraskevi,
GR-153 10, Athens, Greece.
{dfarmak, vangelis, jkoutsi, petasis, costass}@iit.demokritos.gr

Abstract

This paper addresses the problem of Information Extraction (IE) system customization to new domains and extraction needs with the use of PatEdit, an IE Pattern Editor. PatEdit is a human-assisted knowledge engineering tool, that facilitates the production of IE patterns. First, we present the problem of IE system customisation and the use of human assisted knowledge engineering tools. Then, we describe PatEdit with respect to the IE pattern language used and discuss its characteristics that facilitate rapid pattern writing. Finally, the exploitation of PatEdit in two information extraction projects is presented along with our plans for future work.

1. Introduction

Information Extraction (IE) systems fill in predefined data structures (e.g. relational databases) involving a particular event (e.g. company acquisitions) with information they extract from unstructured natural language texts that refer to a particular domain (e.g. financial news). They do not attempt to “comprehend” the texts they process in their entirety, but they simply try to extract from them all the information that is needed to fill in predefined data structures.

Within the Message Understanding Conferences[1] (MUCs), IE is decomposed into the following sub-tasks: *Named Entity* (involving the recognition of specific semantic types of proper names and expressions), *Coreference* (detection of coreferential expressions), *Template Element* (identification of descriptions referring to specific types of entities), *Template Relations* (discovery of semantic relations holding between entities) and the *Scenario Template* a task that uses the results of the preceding sub-tasks for the generation of a template that describes an entire event.

IE systems involve general purpose linguistic processing (e.g. sentence splitting, part of speech tagging) as well as task specific processing for the identification of the elements (phrases, names, expressions) that fill the slots of an event template. Linguistic processing relies on general-purpose resources that do not require painstaking adaptation to new domains or new types of events within a given domain. Task specific processing, on the other hand, involves extraction patterns (or rules or concept nodes depending on the system) and resources (domain specific gazetteers) that must be written anew whenever a new event or domain is dealt with. The creation of extraction patterns for a new domain or event requires domain expertise and awareness of the underlying IE system structure. This renders the customization of IE systems an expensive and time-consuming process.

One of the ways in which researchers have attempted to address the customization bottleneck of existing IE systems is tools that allow domain experts to create extraction patterns fast and easily without requiring knowledge of linguistics or language engineering, as well as tools that facilitate the work of the knowledge engineer in adapting a system to different domains. Section 2 provides more details on these tools and explains the motivation behind the development of PatEdit.

Section 3 presents the Pattern Representation language used by PatEdit, whereas Section 4 presents the main functionalities of PatEdit. Section 5 discusses the use of PatEdit in two different IE projects. Finally, in Section 6 we present our concluding remarks and our plans for future work.

2. Related Work

Tools that allow the customization of IE to new domains can be distinguished in two broad categories depending on the user they address. The first category addresses domain experts, while the second one addresses knowledge engineers and IE system developers.

2.1. Tools for domain experts

Tools for domain experts may require different degrees of user involvement in the system customization task.

An example of a tool that requires minimal user involvement is the template construction interface of the LOLITA IE system (Costantino 1997). Through the LOLITA template construction interface, the domain expert can define new templates using natural language and a number of formal elements for ensuring unambiguous definitions. In this framework, the template definition is comprised of a template name, variables that identify slot names, an enabling condition for the construction of the template, slot names and slot rules. The system translates template definitions into semantic rules that are used by an internal inference engine that conducts IE. It must be noted that this type of customization is viable only for systems that conduct natural language understanding, which involves deep

[1] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

syntactic analysis as well as semantic and pragmatic analysis, rendering them too expensive to build.

The Concept Node Editor developed for the CIRCUS IE system (Lehnert et al. 1993), involves a higher degree of expert user involvement in system customization. Using this editor the domain expert can review concept nodes (equivalent to extraction patterns) that have been constructed by processing a corpus with a system called AutoSlog. The user distinguishes between valid and invalid concept nodes for a specific extraction task. Even though individual experts may not select the same number of concept nodes from a given set, it has been shown that the results obtained by the concept nodes selected by domain experts were comparable to the results produced by concept nodes that had been created manually by researchers.

Tools based on corpus analysis for the construction and validation of extraction patterns requires the highest degree of domain expert involvement. The PROTEUS Pattern Acquisition Tools (Yangarber et al. 1997) assist the user in the creation of patterns by parsing examples that are fed to the tool by the user. The user must associate the elements of the parsed example to predefined event slots for the creation of an extraction pattern and can perform a set of operations on patterns such as generalization of an element or marking an element as optional. Finally the user can test against a corpus if a pattern produces the desired results. A significant advantage of this set of tools is that, given a pattern, the user is not involved in the creation of variant patterns, e.g. produce the passive form of an active pattern, since this is performed automatically by a tool that performs syntactic transformations.

More recent suggestions for IE customization tools view the domain experts as the final users of an IE system and stress their autonomy with the following requirements:

- Assistance in the creation of new event templates and the compilation of relevant corpora (Ciravegna et al. 2001)
- Facilities for tagging corpora and use of machine learning techniques for automatic acquisition of extraction patterns (Catala et al. 2000, Ciravegna et al. 2001)
- Facilities that enable users to tune a system to individual extraction needs, e.g. a user may prefer more recall than precision (Ciravegna et al. 2001)

2.2. Tools for knowledge engineers

A limited number of tools has been developed for rapid system customization by knowledge engineers.

(Doran et al. 1997) describe a graphical debugger for MOP, an IE pattern description language. This tool assists the knowledge engineer in monitoring the results of the application of a MOP IE pattern to a text at different execution stages. Within an interface that exhibits the parts of the text that matched and the results obtained by each one of the components of an extraction pattern, the user can gradually improve extraction patterns or add new ones.

(Ciravegna et al. 2000) propose Pinocchio, an environment for developing and adapting an existing IE system to new domains and languages. Pinocchio presents the knowledge engineer with resources for one language and an extraction task and a set of support tools for resource development and testing. An editor and a compiler are provided for the development of extraction rules. Specialized browsers and tracers for extraction rules and linguistic analysis results are provided for resource testing. This set of tools is claimed to enable system customization to new events and system extension to new languages by only one computational linguist or knowledge engineer.

2.3. The motivation behind PatEdit

System customization is one of the main objectives of our laboratory[2] concerning the development of IE systems and in general language engineering applications. For this purpose, we have developed Ellogon, a text-engineering platform (Petasis et al. 2002), which is equipped with annotation tools, viewers of the results obtained by different processing modules, tools that facilitate the creation of training vectors for machine learning techniques. Ellogon provides infrastructure for managing and exchanging textual data as well as the associated linguistic information, creating, embedding and managing linguistic processing components, as well as facilitating communication among different linguistic components by defining a suitable API. The facilities provided by Ellogon along with the general purpose linguistic processing tools and the machine learning techniques and tools developed in our laboratory facilitate the customisation of IE systems. Our objective is to develop an IE workbench over the Ellogon platform in order to facilitate the development and maintenance of IE systems. In this context, in order to meet the needs of our IE technology, we decided to develop PatEdit. These needs can be summarized in the following:

- Definition of new event templates in a menu driven environment since our IE technology employs pattern matching techniques that do not make use of natural language understanding and deep linguistic analysis
- Addition of new types of information in the pattern representation language we use, so it can support future developments of an existing IE system, e.g. addition of new types of semantic information, new types of operators, as well as new IE systems developed under the same framework
- Fast construction of valid extraction patterns for different domains and events both for domain experts and knowledge engineers

PatEdit is a human-assisted knowledge engineering tool that facilitates the production of IE patterns. It can also be used by domain experts who have minimal knowledge of linguistics and no knowledge of the underlying IE system's function.

3. The Pattern Representation Language

PatEdit has been developed in the context of the MITOS IE system[3] (MITOS-IE). MITOS-IE processes Greek Financial News texts and fills in predefined templates for 10 different financial events (Kopanaki et al. 2001). The system architecture involves Lexical

[2] http://www.iit.demokritos.gr/skel

[3] http://www.iit.demokritos.gr/skel/mitos/index_en.html

Processing (tokenization, part of speech tagging, chunking, shallow parsing), Text Classification, Named Entity Recognition, Name Coreference, Template Element, and Scenario Template. The Scenario Template module uses extraction patterns for the identification of relations between different entities. The extraction patterns employed for filling specific event templates are relational multi-slot patterns. They are written in a domain-independent Pattern Representation Language, which allows them to be insensitive to word and phrase order, which is important for Greek.

The patterns embody a 3-layer structure comprised of basic units, phrase patterns and complete patterns. Basic units include names of template slots to be filled (e.g. BUYER in an Acquisition event), phrase types (e.g. Noun Phrase), grammatical relations (e.g. Subject), lexical information (e.g. lemma, tense, case) and semantic information (e.g. Organization name, monetary expression).

Phrase patterns are comprised of basic units and may carry certain weights depending on how confident the pattern writer is about the results the patterns produce. A phrase pattern may state, for instance, that a BUYER of an Acquisition event is to be found in a noun phrase (NP) that is a subject (PHRASESUBJ), and contains the lemma "company" (TOKLEMMA|company), as well as the name of an Organization (NAMETYPE|ORG). If weights are used in the creation of the pattern file, a weight is added at the beginning of the pattern to show how accurately this pattern is expected to identify a slot-filler. General patterns are assigned low weights, whereas more precise patterns are assigned high weights.

Phrase Pattern examples:

```
BUYER+NP+PHRASESUBJ+TOKLEMMA|company
+NAMETYPE|ORG

W3^BUYER+NP+PHRASESUBJ+TOKLEMMA|company
+NAMETYPE|ORG
```

The conditions expressed by a single phrase pattern may presuppose conditions present in other phrase patterns that belong to the same final or complete extraction pattern. For instance, the condition that a noun phrase must be a subject presupposes that a pattern for a verb phrase exists in the complete extraction pattern and that the conditions imposed by this verb phrase pattern are met. So complete or final extraction patterns consist of phrase patterns that may be associated in some manner and appear in the same sentence.

Final Extraction Pattern examples:

```
TIMEPAST+VP+TOKLEMMA|acquired@
BUYER+NP+PHRASESUBJ+TOKLEMMA|company
+NAMETYPE|ORG@
ACQUIRED+NP+PHRASEDOBJ+NAMETYPE|ORG

W3^TIMEPAST+VP+TOKLEMMA|acquired@
W3^BUYER+NP+PHRASESUBJ+TOKLEMMA|company
+NAMETYPE|ORG@
W3^ACQUIRED+NP+PHRASEDOBJ+NAMETYPE|ORG
```

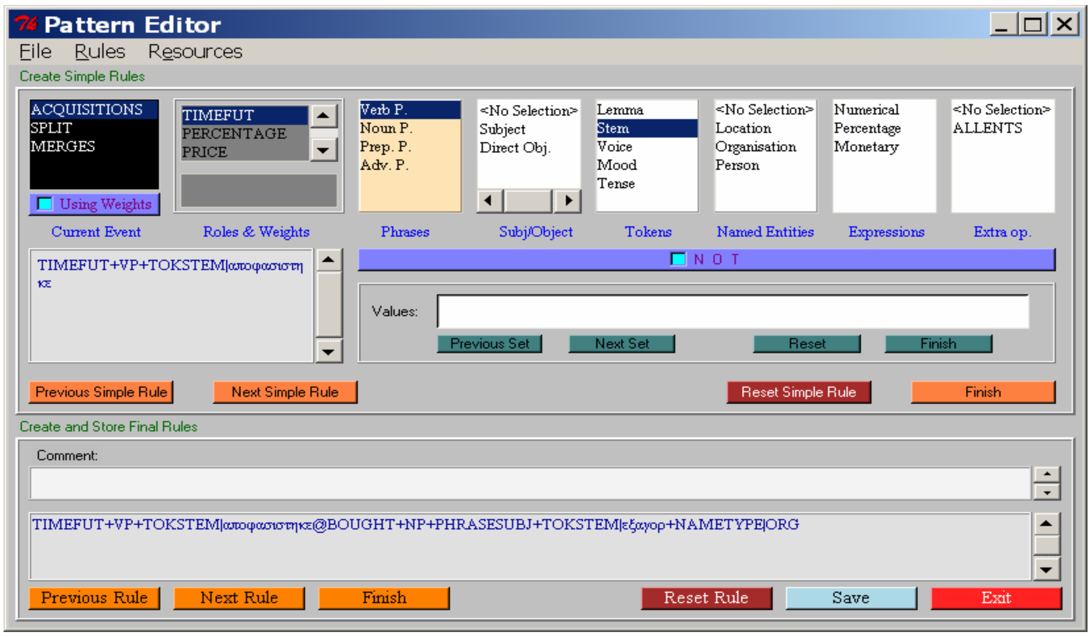
For the extraction of slot fillers complete patterns are matched against an internal representation of the text that is rich with information from previous processing.

The construction of extraction patterns requires, apart from domain knowledge, familiarity with the information existing from previous processing and the Pattern Representation Language. Even when the pattern writer is familiar with the system, minor mistakes like misspellings of reserved words of the Pattern Representation Language may occur, e.g. writing "TOKLEM" instead of "TOKLEMMA". Mistakes in the syntax of patterns may also occur since the form of the complete extraction patterns is rather rigid. Certain constituents need precede others, e.g. verb phrase patterns must precede all other types of phrase patterns in a final rule. Also certain types of features must appear in a certain order to provide valid patterns, e.g. numeric expressions must follow named entities in a phrase pattern. Extraction rules or patterns that contain misspellings or syntax errors are not processed by the IE system. Invalid patterns can be detected only with evaluations of a system's performance and error analysis during an IE system's development circle. Thus detection of invalid patterns must be performed every time new extraction patterns are added or existing patterns are modified making the expansion of the extraction pattern resources and system customization to new events and domains an expensive and time-consuming process.

4. PatEdit Functionalities

In order to accelerate the process of writing valid extraction patterns for new domains and events we have developed PatEdit, a pattern editor that allows fast and easy creation of new patterns, modification of existing ones, and deletion of unwanted ones, as well as definition of new event templates and information to be used in the construction of patterns. PatEdit has been implemented in Tcl/Tk v8.4a2, and the environment vTcl v.1.5.2 has been used in the construction of the GUI. It is part of a larger workbench based on the Ellogon text-engineering platform that supports the development of language engineering applications.

Using PatEdit, the user constructs extraction patterns by clicking specific types of information from selection lists and by assigning values to specific features. Thus the user is no longer burdened with typing commonly occurring constructions of the pattern representation language. Moreover they do not have to check whether the patterns conform to the Pattern Representation Language. The basic units that comprise the patterns are assembled by the tool in the order required by the Pattern Representation Language, irrespectively of the order in which the user has selected or specified them. In case the user attempts to write a pattern that violates important conditions of the Pattern Representation Language, e.g. a type of phrase is not specified in a phrase pattern, they are not allowed to proceed in the creation of an invalid pattern, but they are presented with messages explaining the type of violation encountered and the course of action needed in order to make the pattern conform to the Representation Language. The user is also guided to the creation of pattern files. Thus, the extraction patterns and the pattern files produced with PatEdit are error proof, since PatEdit does not allow invalid pattern constructions and malformed files.



TIMEFUT+VP+TOKSTEM|αποφασιστηκε

TIMEFUT+VP+TOKSTEM|αποφασιστηκε@BOUGHT+NP+PHRASESUBJ+TOKSTEM|εξαγορ+NAMETYPE|ORG

Figure1. A screenshot of the main GUI

4.1.1. PatEdit: the Main GUI

The Main GUI (Figure 1) is the one used for the creation of the extraction patterns. It presents the user with three menus, and two frames: one for the creation of simple phrase rules and one for the creation and storage of final rules.

From the menus the user can perform the following actions:

- Create a new file for saving the rules
- Open existing rule files
- Save an open rule file
- Terminate the application
- Open a rule file for viewing and selecting existing rules for editing
- Add new rules to an existing rules file
- Open the Resources GUI that allows the configuration of the tool to handle new types of events

The first frame named "Create simple rules" presents the user with all the building blocks required for the construction/creation of a phrase pattern, a viewer that allows them to see the pattern under construction and buttons that allow them to accept, reset or edit it.

The selection lists appearing at the top of the frame embody the types of information required for the creation of simple phrase patterns. The first one is the Event selection list, exhibiting all the events the tool can be used to write extraction rules for. Whenever an event is selected from this list, the event slots or roles associated with it appear in the next selection list. If the Acquisition event is selected, for instance, the relevant event slots "Buyer", "Acquired", "Past", "Present", "Future", "Percentage" and "Amount" appear in the next selection list. Six more selection lists appear after the first two presenting types of information required for the creation of patterns:

- Phrases
- Grammatical Relations
- Lexical Information
- Named Entity Types
- Expression Types
- System Operators

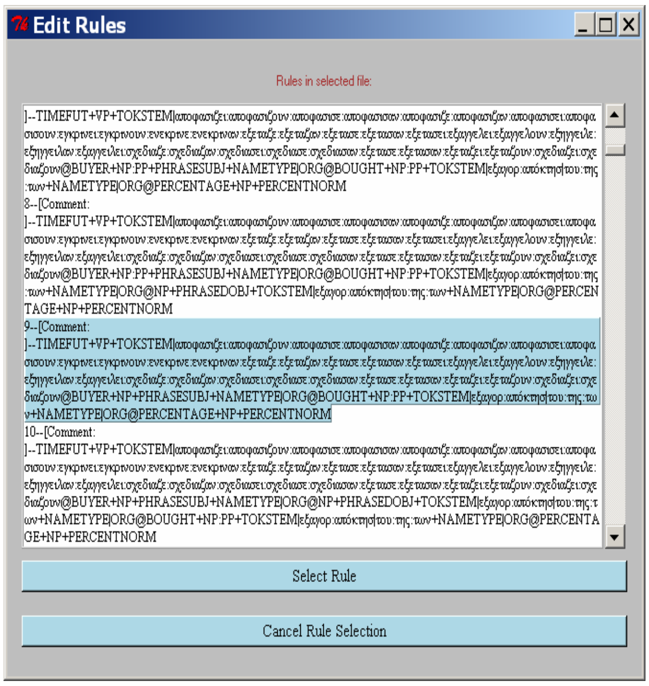
For the creation of a simple phrase pattern, the user can select one or more types of phrases from the phrase selection list and any type of features that are not mutually exclusive from the next selection lists. A mutually exclusive pair is the subject and object in the Grammatical Relations list, since the same phrase cannot be subject and object of the same verb phrase. The Lexical Information list contains features like "lemma", "voice", "stem" for which, when selected, specific sets of required or alternative values must be typed by the user in the values text entry box. For instance, if the user selects "voice", a value for voice, e.g. "active" must be typed in the value text entry box. From the Named Entity Types list (e.g. Organization, Location) selection of only one named entity type is allowed, whereas the Expression Types list allows the selection of more than one expression types (e.g. temporal expression, monetary expression). The System Operators list is comprised of only one operator for the time being, the ALLENTS operator employed whenever we want to specify that all the entities within a phrase that can act as slot fillers for a specific event slot must fill that slot.

Underneath the selection lists, two selection buttons appear: the first one is for use of weights in the construction of the patterns the next one is for the negation operator. Ticking the weights selection button when starting a new rules file, the user activates a list of weights from which he/she can choose a confidence weight for every phrase pattern. Ticking the NOT selection button the Phrase, Grammatical Relations, Lexical Information, Named Entity Type and Expression lists are marked in blue and the user must select from these lists which characteristics the phrase pattern should not have.

The second frame titled “Create and Store final rules” consists of a text entry box for the addition of comments to the rule, a final rule viewer that shows the updated final pattern whenever a new simple phrase rule has been created, and a set of buttons for viewing previous patterns, accepting a final rule, resetting a final rule, and saving a file with extraction patterns.

4.1.2. Handling Existing Rule Files

PatEdit enables the handling of rule files comprised of several patterns by giving the user the chance to open the file in a separate viewer (Figure 2). In this viewer all patterns in the file appear and the user can select the pattern they wish to modify. When a pattern is selected from the viewer it is loaded to the main GUI and can be edited. Using this option the user has a complete picture of their work and can spot and load easily the patterns they wish to revise.

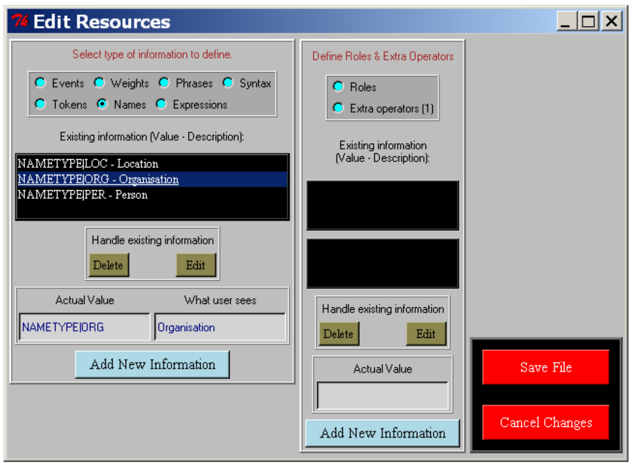


```
]--TIMEFUT+VP+TOKSTEM|αποφασιζει:αποφασιζουν:αποφασισε:αποφασισαν:αποφασιζε:αποφασιζαν:αποφασισει:αποφα
σισουν:εγκρινει:εγκρινουν:ενεκρινε:ενεκριναν:εξεταζε:εξεταζαν:εξετασε:εξετασαν:εξετασει:εξαγγελει:εξαγγελουν:εξηγγειλε:
εξηγγειλαν:εξαγγειλει:σχεδιαζε:σχεδιαζαν:σχεδιασει:σχεδιασε:σχεδιασαν:εξετασε:εξετασαν:εξεταζει:εξεταζουν:σχεδιαζει:σχε
διαζουν@BUYER+NP:PP+PHRASESUBJ+NAMETYPE|ORG@BOUGHT+NP:PP+TOKSTEM|εξαγορ:απόκτησ|του:της
:των+NAMETYPE|ORG@PERCENTAGE+NP+PERCENTNORM
8--[Comment:
]--TIMEFUT+VP+TOKSTEM|αποφασιζει:αποφασιζουν:αποφασισε:αποφασισαν:αποφασιζε:αποφασιζαν:αποφασισει:αποφα
σισουν:εγκρινει:εγκρινουν:ενεκρινε:ενεκριναν:εξεταζε:εξεταζαν:εξετασε:εξετασαν:εξετασει:εξαγγελει:εξαγγελουν:εξηγγειλε:
εξηγγειλαν:εξαγγειλει:σχεδιαζε:σχεδιαζαν:σχεδιασει:σχεδιασε:σχεδιασαν:εξετασε:εξετασαν:εξεταζει:εξεταζουν:σχεδιαζει:σχε
διαζουν@BUYER+NP:PP+PHRASESUBJ+NAMETYPE|ORG@BOUGHT+NP:PP+TOKSTEM|εξαγορ:απόκτησ|του:της
:των+NAMETYPE|ORG@NP+PHRASEDOBJ+TOKSTEM|εξαγορ:απόκτησ|του:της:των+NAMETYPE|ORG@PERCEN
TAGE+NP+PERCENTNORM
9--[Comment:
]--TIMEFUT+VP+TOKSTEM|αποφασιζει:αποφασιζουν:αποφασισε:αποφασισαν:αποφασιζε:αποφασιζαν:αποφασισει:αποφα
σισουν:εγκρινει:εγκρινουν:ενεκρινε:ενεκριναν:εξεταζε:εξεταζαν:εξετασε:εξετασαν:εξετασει:εξαγγελει:εξαγγελουν:εξηγγειλε:
εξηγγειλαν:εξαγγειλει:σχεδιαζε:σχεδιαζαν:σχεδιασει:σχεδιασε:σχεδιασαν:εξετασε:εξετασαν:εξεταζει:εξεταζουν:σχεδιαζει:σχε
διαζουν@BUYER+NP+PHRASESUBJ+NAMETYPE|ORG@BOUGHT+NP:PP+TOKSTEM|εξαγορ:απόκτησ|του:της:τω
ν+NAMETYPE|ORG@PERCENTAGE+NP+PERCENTNORM
10--[Comment:
]--TIMEFUT+VP+TOKSTEM|αποφασιζει:αποφασιζουν:αποφασισε:αποφασισαν:αποφασιζε:αποφασιζαν:αποφασισει:αποφα
σισουν:εγκρινει:εγκρινουν:ενεκρινε:ενεκριναν:εξεταζε:εξεταζαν:εξετασε:εξετασαν:εξετασει:εξαγγελει:εξαγγελουν:εξηγγειλε:
εξηγγειλαν:εξαγγειλει:σχεδιαζε:σχεδιαζαν:σχεδιασει:σχεδιασε:σχεδιασαν:εξετασε:εξετασαν:εξεταζει:εξεταζουν:σχεδιαζει:σχε
διαζουν@BUYER+NP+PHRASESUBJ+NAMETYPE|ORG@NP+PHRASEDOBJ+TOKSTEM|εξαγορ:απόκτησ|του:της:τ
ων+NAMETYPE|ORG@BOUGHT+NP:PP+TOKSTEM|εξαγορ:απόκτησ|του:της:των+NAMETYPE|ORG@PERCENTA
GE+NP+PERCENTNORM
```

Figure 2. A screenshot of the Edit Rules Function

4.1.3. The Edit Resources GUI

In the Edit Resources GUI (Figure 3), the user can configure the tool to handle new types of events, and manage the main GUI. From the first frame of Edit Resources, the user can handle the information types appearing in the selection lists (e.g. Events, Phrases) of the main GUI. Thus information types and their associated values in the main GUI and the Pattern Representation Language can be modified or discarded and new types can be added. In the second frame of Edit Resources names of roles (i.e. names of slot fillers for specific events) and operators can be added or modified. When the user saves the file by pressing the “Save File” button, they are warned that the application will terminate and must be restarted for changes to take effect.


Edit Resources
Select type of information to define.
Handle existing information
Delete
Edit
Add New Information
Define Roles & Extra Operators
Roles
Extra operators (1)
Existing information (Value - Description):
Handle existing information
Delete
Edit
Actual Value
Add New Information
Save File
Cancel Changes

Select type of information to define: Events Weights Phrases Syntax Tokens [✓ Names] Expressions

Existing information (Value - Description):

NAMETYPE|LOC - Location
NAMETYPE|ORG - Organisation
NAMETYPE|PER - Person

Actual Value: NAMETYPE|ORG
What user sees: Organisation

Figure3. The Edit Resources GUI

5. Exploitation of PatEdit

In the context of the MITOS-IE system, PatEdit has been used to include new financial events for Greek texts. It has contributed to the rapid development of valid extraction patterns by users familiar with the Pattern Representation Language as well as users who were unfamiliar with it. Extraction rule files have been constructed for 10 financial event templates in total (Acquisitions, Merges, Split, Capital Stock Increase, Introduction to the Athens Stock Exchange Market, Profits, Sales, Free Share Distribution, Turnover, and Dividend Yield). Extraction patterns have been written without the use of PatEdit for the first five events over a period of 4 months, whereas development of extraction patterns for the next 5 events with the use of PatEdit took only 1 month.

Moreover, PatEdit is currently being used in the CROSSMARC[4] project, which develops IE technology for multilingual e-retail product comparison. The type of text we deal with in this project is hypertext, which is quite different from flat text news items, and the first domain is Computer Goods. Given the Edit Resources GUI we are able to configure the tool for the new domain and text type with the addition of new events (e.g. Laptop Offer), new slot names associated with them (e.g. IS_MODEL) and new types of information (e.g. title, paragraph, list item).

6. Concluding Remarks and Future Development

We have presented PatEdit, an IE system adaptation tool that allows fast construction and editing of information extraction pattern files conforming to a certain Pattern Representation Language.

PatEdit forms a component of the IE technology workbench we are developing in our laboratory over the Ellogon text engineering platform. This workbench involves general purpose language processing tools, annotation tools (for the creation of the necessary training and testing corpus), machine learning tools, facilities for creating training vectors for machine learning, named entity recognition tools (gazetteer lookup, parser), a pattern representation language for the creation of IE patterns, a pattern matcher, as well as

[4]http://www.iit.demokritos.gr/skel/CROSSMARC

the PatEdit. We exploited these tools and facilities for the development of the MITOS-IE system (extraction of information from financial news), which is currently being customized in a different application domain (extraction of products characteristics from web pages) in the context of the CROSSMARC project.

PatEdit can be easily configured for new event templates as well as for variations (presentation of new types of information and new operators) in the Pattern Representation Language. It has been used successfully for adapting MITOS-IE system to new financial events and it is being used in CROSSMARC for the construction of extraction patterns for descriptions of computer goods from Web pages.

The provision of a facility for testing a single extraction pattern against a corpus would be very useful for the patterns author. We intend to incorporate this facility in PatEdit since inspection of results in a large corpus of several texts facilitates the construction of patterns that do not overgenerate, promoting system accuracy. Moreover, comparison of system extraction results to results provided by manual annotation facilitates the construction of general patterns that promote system recall. Currently the user can view and compare extraction results using Ellogon built-in tools but the results come from all the patterns present in a file for a specific event and detecting patterns that have produced erroneous results can be a time consuming process depending on the size of the pattern file.

The incorporation of domain specific resources like term dictionaries and thesauri is also a desired feature that we plan to incorporate in PatEdit. Allowing the user to specify the type of dictionary to use and insert synonyms easily in extraction patterns is our second aim for further development.

References

Català, N., Castell, N., Martín, M. (2000) ESSENCE: a Portable Methodology for Acquiring Information Extraction Patterns. In *Proceedings of 14th European Conference on Artificial Intelligence* (ECAI-2000) (pp. 411--415). Berlin: IOS Press.

Ciravegna, F., Petrelli, D. (2001) User Involvement in Customizing Adaptive Information Extraction: Position Paper. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining* held in conjunction with the 17th International Conference on Artificial Intelligence (IJCAI-01), (http://www.smi.ucd.ie/ATEM2001/proceedings/ciravegna-position-atem2001.pdf).

Ciravegna, F., Lavelli, A., Satta, G. (2000) Bringing Information Extraction out of the Labs: the Pinocchio Environment. In *Proceedings of the 14th European Conference on Artificial Intelligence* (pp. 416--420). Berlin: IOS Press.

Costantino, M. (1997) Financial Information Extraction Using Pre-defined and User-definable templates in the LOLITA System. Phd Thesis, University of Durham.

Doran, C., Niv, M., Baldwin, B., Reynar, J.C. and B. Srinivas (1997). Mother of PERL: A Multi-tier Pattern Description Language. In *Proceedings of the Lexically-Driven Information Extraction Workshop* (pp. 13--22). Universita di Roma: Dept. of Computer Science.

Kopanaki, E., Karkaletsis, V., Spyropoulos, C.D., Avradinis, N., Fakotakis, N., Kalamboukis, Th., Kladis, B., Lazarou, Y., Panayiotopoulos, Th.., Spinellis, D. (2001). MITOS: An Integrated Web-based System for Information Management. In *Proceedings of the 8th Panhellenic Conference on Informatics*, (pp. 328--337). Nicosia, Cyprus: Livanis.

Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F., Dolan, C., Goldman, S. (1993). UMASS/HUGHES: Description of the Circus System Used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference* (pp. 277--291). Baltimore: Morgan Kaufmann.

Petasis, G., Karkaletsis, V., Paliouras, G., Androutsopoulos, I., and Spyropoulos, C.D. (2002) “Ellogon: A New Text Engineering Platform”. In Proceedings of the *Language Resources and Evaluation Conference (LREC-2002).* Las Palmas, Spain.

Yangarber, R., Grishman, R. (1997). Customization of Information Extraction Systems. In *Proceedings of the Lexically Driven Information Extraction Workshop* (pp. 1--11). Universita di Roma: Dept. of Computer Science.