A pattern extraction workbench combining multiple linguistic levels

Magnus Merkel & Andreas Lange

Department of Computer and Information Science,
Linköping University
S581 83 Linköping, Sweden,
{mme,anlan}@ida.liu.se

Abstract

In this paper an interactive pattern extraction workbench, I*Pex, is presented. The workbench comes in a graphical environment and is designed to be used in an incremental and interactive fashion with the user. Patterns can be constructed to work in combination involving specifications on several linguistic levels simultaneously, from the character level using regular expressions, parts of speech and dependency relations to semantic roles. The input text format is based on XCES XML format.

1. Introduction

A notorious problem for customizing information extraction applications regards the extent on which patterns can be reused or easily adapted for new domains and new tasks. As designing patterns for semantic annotation and extraction is a tedious task, attempts at making this process more efficient and easier to complete is a necessary step towards better IE applications.

Several attempts at finding a flexible and generic approach to address the problems of creating patterns for IE have been made. Within the TIPSTER framework the Common Pattern Specification Language (CPSL) was developed and used in applications such as Doug Appelt's TextPro (Appelt & Martin 1999). In work with the GATE architecture a Java Pattern Annotation Engine (JAPE) was developed and used in several IE systems, for example SOCIS (Cunningham et al. 2000). JAPE is built on CPSL with some modifications. Both CPSL and JAPE handle pure text files using rules based on regular expressions. Interesting work on making the process of customisation in IE systems more efficient have been made in the Proteus project where example-based strategies for pattern building were used (Yangarber & Grishman 1997). The interactive approach to pattern extraction described in this paper provides the user with the possiblity to test and evaluate patterns giving the user immediate visual feedback on the performance of existing and new patterns for a particular task.

2. I*Pex

In this paper we present a pattern extraction workbench called I*Pex, which is part of a series of tools that share the features of being highly interactive and intended to use in an incremental fashion where each step can be tested and verified in user-friendly environments. Other tools in the series so far are mostly concerned with multilingual data extraction through word alignment techniques (cf. Ahrenberg et al., 2003, Merkel et al. 2003). The tools all share the feature that input documents have been analysed by a functional dependency grammar parser, in this case Connexor's Machinese Syntax (Tapanainen & Järvinen 1997), and converted into an XML notation partially based on the XCES Corpus Encoding Standard (XCES 2000). However, it is not necessarily restricted to one type of grammatical analysis tool as long as the XML mark-up adheres to the XML notation specified in the DTD.

In the following we make an attempt at describing the functionality of I*Pex as well as providing some examples of how it can be used. The tool has primarily been applied on Swedish text materials, but there is nothing in the architecture that prevents it from being applied to other languages as long as the XML-mark-up complies with the document type definition. The first stage in working with I*Pex concerns the tagging of the input documents, which is performed by Connexor's Machinese Syntax parser. The tagging information from Machinese Syntax is relatively rich, involving not only stemming and POS information but also morphological features, syntactic functions and functional dependencies. The XML format that is used is to some extent based on the XCES standard, which gives the following analysis for a sentence such as *Senast den 16 december skickar skattemyndigheten ut slutskattebeskedet.*(Eng. Transl: *At the latest December 16 the tax authorities will send out the final tax result.*:

```
<s id="s3"> <w id="w29" relpos="1" base="sen"
func="advl" fa="&gt;5" stag="AH" pos="ADV"
msd="&lt;Sup&gt;"> Senast</w>
  <w id="w30" relpos="2" base="den" func="det"
fa="&gt;4"  stag="&gt;N" pos="DET"
msd="SG+NOM">  den</w>
<w id="w31" relpos="3" base="16" func="attr"
fa="&gt;4" stag="&gt;N" pos="NUM"
msd="NOM+&lt;Card&gt;">16</w>
<w id="w32" relpos="4" base="december"
func="advl" fa="&gt;5" stag="NH" pos="N"
msd="SG+NOM"> december </w>
<w id="w33" relpos="5" base="skicka" func="main"
fa="&gt;0"  stag="MV" pos="V" msd="PRES">
skickar</w>
<w id="w34" relpos="6" base="skatte#myndighet"
func="subj" fa="&gt;5" stag="NH" pos="N"
msd="SG+NOM"> skattemyndigheten</w>
<w id="w35" relpos="7" base="ut" func="advl"
fa="&gt;5" stag="AH" pos="ADV">ut</w>
<w id="w36" relpos="8" base="sluts#katte#besked"
func="obj" fa="&gt;5" stag="NH" pos="N"
msd="SG+NOM"> slutskattebeskedet</w>
<w id="w37" relpos="9" base="." stag="INTERP"
pos="INTERP" msd="Period">.</w></s>
```

Figure 1. XML mark-up after syntactic tagging.

I*Pex can be seen as a task-oriented semantic tagger in that it can assign semantic properties to portions of text according to the patterns constructed by the user. In I*Pex there are two principal modes of working with patterns:

1. bottom up (starting with primitive patterns to more complex patterns);
2. top down (scenario templates that are instantiated using information assigned by patterns in the bottom-up approach).

In the first mode simple patterns, for example identifies all nouns starting with upper-case letters, can be constructed and named-entity-like patterns distinguishing between different types of name descriptions could then be built using the previous patterns. In the second mode, I*Pex will start out with a number of scenario templates that are instantiated using the original linguistic information and the semantic properties assigned by the patterns specified in mode 1.

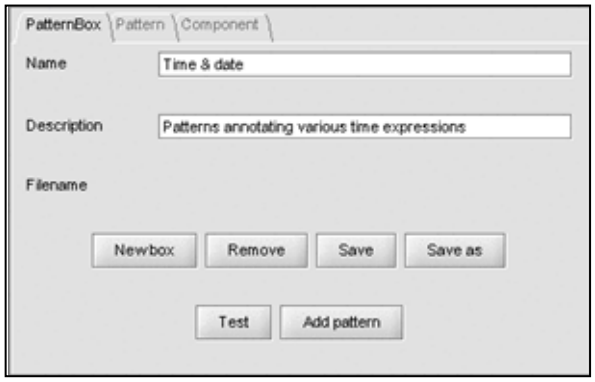

Figure 2 Pattern box specifications

The graphical workspace of the workbench is divided into a number of components: a document browser where the source documents can be viewed; a pattern box window where the user can specify and test individual patterns and pattern boxes, and a graphical display of the results of applying the patterns. In Figure 2 and 3 a pattern box "Time & date" is illustrated where patterns for extracting time and date expressions are specified.

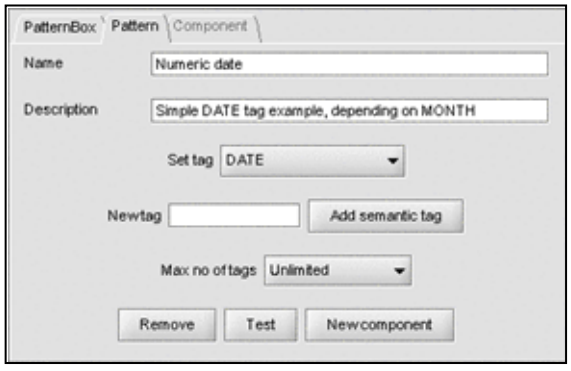

Figure 3 Setting up a pattern for DATE

Normally an I*Pex user loads a project, consisting of XML marked-up text and several *pattern boxes*. Some of the pattern boxes represent reusable basic building blocks handling expressions, such as generic time and date recognition, while others are specific to the task in question. A pattern box is stored as an XML file and contains several annotation patterns.

Each pattern produces a specific annotation. Annotation patterns are further divided into *search components*. A component searches on multiple levels in the text, word by word. Complex interactions between searches are handled by chaining components. A simple pattern may consist of only one component, for example one trying to find all nouns starting with an upper-case letter. A higher level pattern might begin with a component that tries to find a paragraph based on previous annotations. It can then continue with components that only search within specific sentences in this paragraph. In this way complex searches are nested giving the user a very powerful search tool, but at the same time providing the possibility to reuse parts of a complex search later on.

In Figure 4 below, a component for finding DATE expressions is defined. The user has previously defined a MONTH tag using a gazetteer holding month names which is reused with a regular expression "[1-9]|([1,2][0-9])|30|31"on the word level that finds numbers between 1 and 31. There is also an additional criterion in the POS field, requiring that the number should be tagged as a numeral. On the left hand side it is possible to determine various connections between components, such as range (sentences, paragraphs) and what to do with items that match the search conditions. In this particular example, the numerical expression (1-31) has to be connected to a MONTH expression and be placed either in the position exactly before or after the MONTH expression. There are various other possibilities to set ranges and directions. The search criterion could be very complex involving several components tied together in various ways. The result of the complex pattern in Figure 4 is that expressions like "30 september" and "14 april" are assigned the semantic tag DATE.

Figure 4 The Component specification dialog box where ranges and search criteria can be specified, as well as combining pattern components to handle more complex specifications

A user may either choose to run all patterns, the patterns in a specific pattern box or one selected pattern. Before a search is executed the relevant patterns are sorted with regard to their annotational dependencies and then run in order. The components are compiled to handle matching efficiently. A finite state machine runs each

pattern, selects components, sets up search ranges and runs iterators.

The second approach, to run patterns in top down mode in order to do the traditional IE approach of filling scenario patterns is also supported in I*Pex. For example, given that a user wants to pinpoint everything in the text material that describes events involving sending items from somebody to somebody at certain times. A scenario template for such events will involve assigning semantic roles for SENDER, RECEIVER, OBJECT (being sent), TIME, START and DESTINATION for example. The template filling is then performed by placing constraints on the assignment of these roles using both syntactic and semantic features specified by the more primitive patterns described above. An example of such a role is given below in Figure 5 where I*Pex has identified the fact that 'the tax authorities' (*skattemyndigheten*) is the sender, that 'the final tax results' (*slutskattebeskedet*) is what is being sent, and that the time for this event is 'December 16'.

3. Summary of features

Below a number of the specific features of the I*Pex workbench are summarized:

- Operates on syntactically annotated source documents in XML, partially based on the XCES specification with information on word form, base form, POS, morpho-syntactic features, syntactic function and dependency relations.
- Ability to specify patterns that operate on a combination of linguistic levels, for example, by combining information on parts-of-speech, morpho-syntactic features with syntactic function, and the user could easily specify a pattern that locates a noun in the definite plural which in turn functions as a direct object of a specified verb.
- Patterns are specified in a graphical interface, where lower-level patterns can be combined by manipulating objects graphically.
- Patterns related to each other can be stored in pattern boxes for reuse in other domains and projects.
- Individual patterns or collections of patterns (pattern boxes) can be tested on the fly and results are displayed graphically in the workbench environment.
- Annotations created by I*Pex are stored as values of a semantic attribute in a stand-off XML representation with pointers to the XML source file holding the original syntactic mark-up.
- Ranges for complex patterns can be specified for different purposes; some patterns only operate within a construction (e.g. a noun phrase) others within a sentence or within a paragraph.
- Gazetteer data can be loaded and utilised within the patterns.

The system has been developed using iterative development methods. The first version aimed at handling tagged text and perform simple searches. In the second version the search patterns were divided into components. The third version took care of different search ranges and made it possible to combine pattern components more efficiently. The fourth version is the current one. The next planned version includes an interactive search progress viewer, similar to a programming environment's debugger. This will give the user the possibility to follow how the search progresses through the text, to see what matches and what does not. Such functionality will give the user a better understanding of how different settings affect the search and insights on how to improve patterns, tweaking regular expressions and adapting components to the text and the particular task.

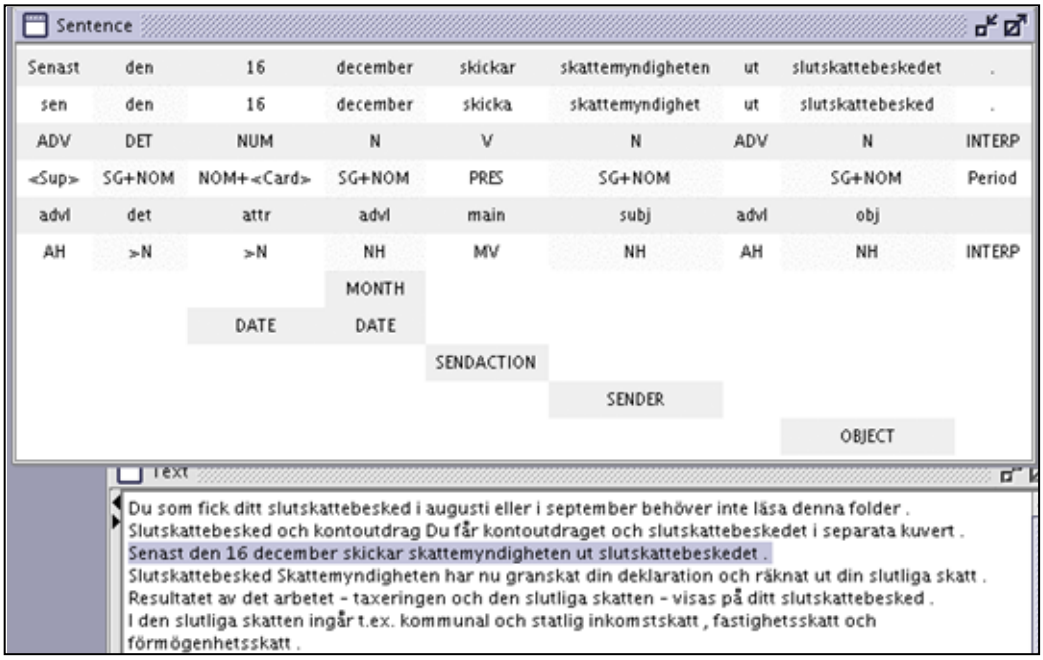

Figure 5. An annotated sentence in the graphical interface where semantic roles for DATE (*16 december*), SENDER (*skattemyndigheten*) and OBJECT (*slutskattebeskedet*) have been assigned by I*Pex. The six lines at the top are the original mark-up given by the tagger and the ones below have been added by I*Pex.

4. Acknowledgements

This work has been supported by The Swedish Agency for Innovation Systems, Vinnova, within the project ”Multimodal interaction for Information Systems”.

5. References

Appelt, D. and D. Martin. 1999. Named entity extraction from speech: Approach and results using the TextPro system. In *Proceedings of the 1999 DARPA Broadcast NEWS Workshop*, 1999, 51–54; www.ai.sri.com/~appelt/SRIIENE.HTM.

Ahrenberg, L., M. Merkel & M. Petterstedt. 2003. Interactive Word Alignment for Language Engineering. Accepted for publication as project note at The 11th Conference of the European Chapter of the Association for Computational Linguistics April 12-17, 2003 Agro Hotel, Budapest, Hungary (EACL-2003).

Cunningham, H., D. Maynard, & V. Tablan. 2000. JAPE: a JAVA Annotation Patterns Engine. Research Memo CS-00-10, Department of Computer Science, University of Sheffield.

Merkel, M., M. Petterstedt & L. Ahrenberg. 2003. Interactive Word Alignment for Corpus Linguistics. Accepted for publication in Proceedings of Corpus Linguistics 2003. UCREL Technical Paper No 16.

Tapanainen, P. and T. Järvinen, 1997. A non-projective dependency parser. *Proceedings of the 5th Conference on Applied Natural Language Processing* (ANLP’97).

XCES Corpus Encoding Standard for XML, 2000, Vassar College, Department of Computer Science, Poughkeepsie NY, http://www.cs.vassar.edu/XCES/.

Yangarber, R. and R. Grishman. 1997. Customization of Information Extraction Systems. In *Proceedings of the International Workshop on Lexically Driven Information Extraction*, Frascati, Italy.

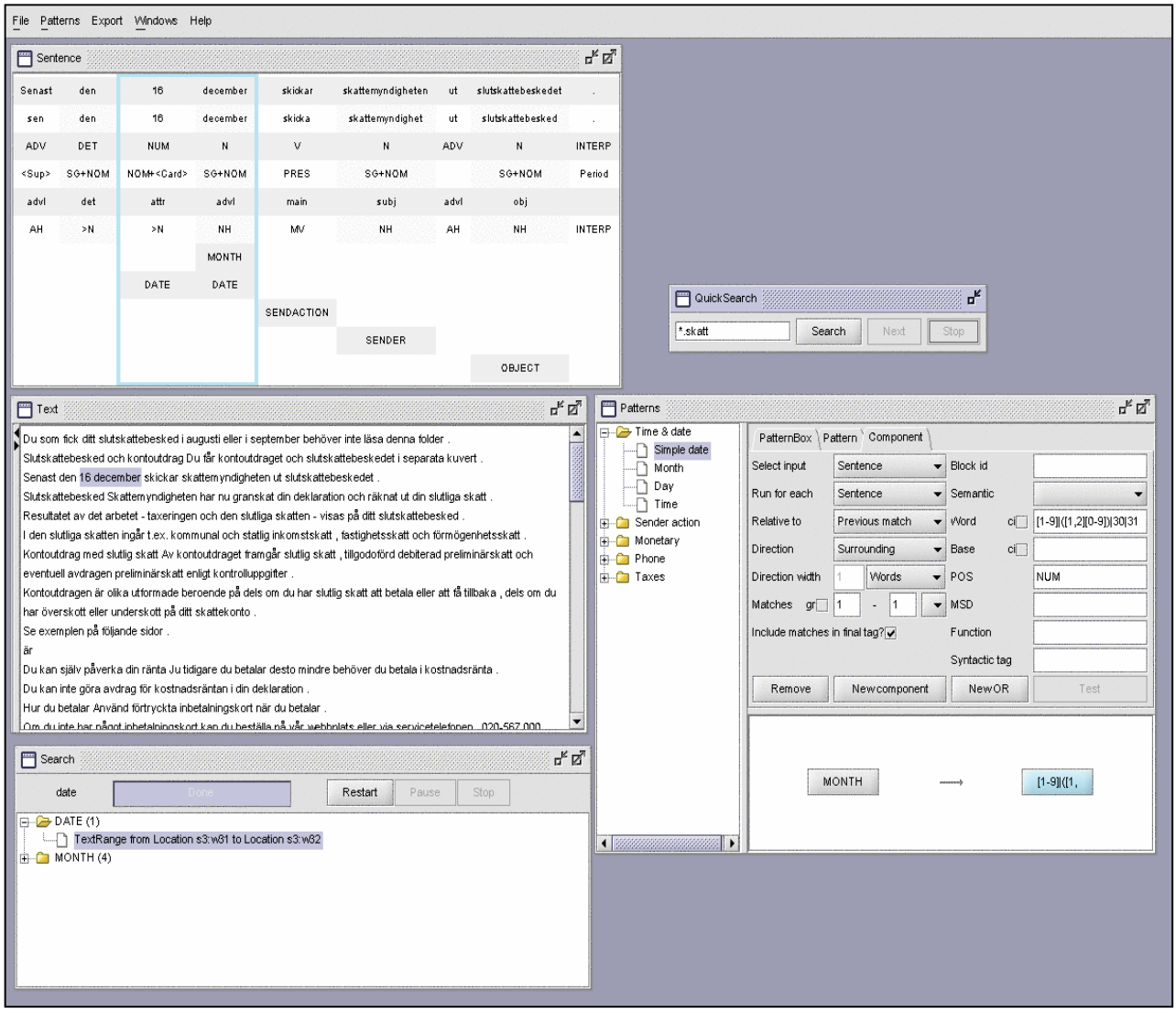

Figure 6. An overview of the I*Pex graphical environment including the Pattern area, the text area, search results and the annotated active Sentence area at the top.