A Framework for Semi-Automatic Development of Rule-based Information Extraction Applications

Peter Kluegl and **Martin Atzmueller** and **Tobias Hermann** and **Frank Puppe**
Department of Computer Science, University of Würzburg, Germany
{pkluegl, atzmueller, hermann, puppe}@informatik.uni-wuerzburg.de

Abstract

For the successful processing and handling of (large scale) document collections, effective information extraction methods are essential. This paper presents a framework for the semi-automatic development of rule-based information extraction applications based on the TEXTMARKER language utilizing machine learning methods. We describe the approach in detail and present the TEXTRULER system as an implementation of the proposed approach.

1 Introduction

Effective methods for information extraction are essential for the (large scale) processing and handling of textual data. In general, information extraction aims to locate specific items in (unstructured) textual documents, e.g., as a first step for more semantic analysis, or for structured data acquisition from text. There exist a variety of automatic methods for information extraction, however, there are also other approaches, e.g., rule-based methods. An implementation of the latter is provided by the TEXTMARKER system wich requires knowledge acquisition. For supporting the developer during rule acquisition step, semi-automatic methods are key techniques.

In this paper, we describe a knowledge-engineering approach incorporating rule-learning methods: For rapid rule capture and prototyping, several rule-learning methods can be applied for acquiring a set of rules that can then be refined later at each level. Machine-learning techniques are applied for acquiring slot and template filler rules for supporting the knowledge engineer when building the set of information extraction rules. The framework supports several machine learning approaches; currently, there are four methods based on the idea of filling single or multi-slot templates specifying the required information. The rules are then either learned, e.g., in a top-down or bottom-up covering manner. The methods are targeted at the TEXTMARKER rule formalization language; TEXTMARKER embeds the proposed framework for rapid rule acquisition using machine-learning techniques.

As an extension of TEXTMARKER, we present the TEXTRULER system as a prototypical implementation of the approach. So far, the approach has been evaluated in several case studies, for example in the medical domain.

The rest of the paper is structured as follows: Section 2 gives a short overview of the TEXTMARKER system and section 3 introduces the semi-automatic development. Then, section 4 concludes with a summary and points at interesting directions for future work.

2 The TEXTMARKER System

The TEXTMARKER system [Atzmueller *et al.*, 2008] is an open source tool[1] for the development of rule-based information extraction applications. The development environment is based on the DLTK[2] framework. It supports the knowledge engineer with a full-featured rule editor, components for the explanation of the rule inference and a build process for generic UIMA Analysis Engines and Type Systems [Ferrucci and Lally, 2004]. Therefore TEXTMARKER components can be easily created and combined with other UIMA components in different information extraction pipelines rather flexibly.

TEXTMARKER applies a specialized rule representation language for the effective knowledge formalization: The rules of the TEXTMARKER language are composed of a list of rule elements that themselves consists of four parts: The mandatory matching condition establishs a connection to the input document by referring to an already existing concept, respectively annotation. The optional quantifier defines the usage of the matching condition similar to regular expressions. Then, additional conditions add constraints to the matched text fragment and additional actions determine the consequences of the rule. Therefore, TEXTMARKER rules match on a pattern of given annotations and, if the additional conditions evaluate true, then they execute their actions, e.g. create a new annotation. If no initial annotations exist, for example, created by another component, a scanner is used to seed simple token annotations contained in a taxonomy.

The TEXTMARKER system provides unique functionality that is usually not found in similar systems. The actions are able to modify the document either by replacing or deleting text fragments or by modifying the view on the document. In this case, the rules ignore some annotations, e.g. HTML markup, or are executed only on the remaining text passages. The knowledge engineer is able to add heuristic knowledge by using scoring rules. Additionally, several language elements common to scripting languages like conditioned statements, loops, procedures, recursion, variables and expressions increase the expressiveness of the language. Rules are able to directly invoke external rule sets or arbitrary UIMA Analysis Engines and foreign libraries can be integrated with the extension mechanism for new language elements.

[1] The source code of the TEXTMARKER project is available at https://sourceforge.net/projects/textmarker/

[2] http://www.eclipse.org/dltk/

3 Semi-Automatic Development

In this section, the semi-automatic development of TEXTMARKER rules is discussed in detail. In the following, we present a comprehensive process model, interesting methods for the automatic acquisition of information extraction rules and the current prototype of the system.

3.1 Process Model

Using the knowledge engineering approach, a knowledge engineer normally writes handcrafted rules to create a domain dependent information extraction application, often supported by a gold standard. When starting the engineering process for the acquisition of the extraction knowledge for possibly new slot or more general for new concepts, machine learning methods are often able to offer support in an iterative engineering process. A conceptual overview of the proposed process model for the semi-automatic development of rule-based information extraction applications is provided in figure 1.

First, a suitable set of documents that contain the text fragments with interesting patterns needs to be selected and annotated with the target concepts. Then, the knowledge engineer chooses and configures the methods for automatic rule acquisition to the best of his knowledge for the learning task: Lambda expressions based on tokens and linguistic features, for example, differ in their application domain from wrappers that process generated HTML pages.

Furthermore, parameters like the window size defining relevant features need to be set to an appropriate level. Before the annotated training documents form the input of the learning task, they are enriched with features generated by the partial rule set of the developed application. The result of the methods, that is the learned rules, are proposed to the knowledge engineer for the extraction of the target concept.

The knowledge engineer has different options to proceed: If the quality, amount or generality of the presented rules is not sufficient, then additional training documents need to be annotated or additional rules have to be handcrafted to provide more features in general or more appropriate features. Rules or rule sets of high quality can be modified, combined or generalized and transfered to the rule set of the application in order to support the extraction task of the target concept. In the case that the methods did not learn reasonable rules at all, the knowledge engineer proceeds with writing handcrafted rules.

Having gathered enough extraction knowledge for the current concept, the semi-automatic process is iterated and the focus is moved to the next concept until the development of the application is completed.

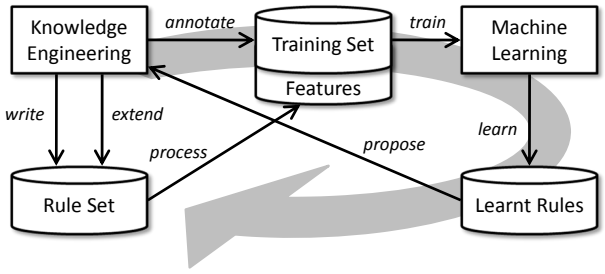

Figure 1: Process model for a semi-automatic development of rule-based information extraction applications.

3.2 Methods

In order to choose appropriate algorithms from the variety of different machine learning techniques for information extraction applications, the following important criteria need to be considered:

- The document type the system operates on.
- Supervised or unsupervised learning strategy.
- Black-box or white-box models.
- Available description of the algorithm.

Since the methods are used in cooperation with the knowledge engineer, rule-based supervised white-box methods fit the described process model best. The goal is to assemble a heterogenous set of methods with techniques for different document types, different learning strategies (e.g. top-down vs. bottom-up) and different rule-representations. The following learning methods all utilize annotated training documents, and have been chosen for further investigation:

BWI

BWI (Boosted Wrapper Induction) [Freitag and Kushmerick, 2000] uses boosting techniques to improve the performance of simple pattern matching single-slot boundary wrappers (*boundary detectors*). Two sets of detectors are learned: the "fore" and the "aft" detectors. Weighted by their confidences and combined with a slot length histogram derived from the training data they can classify a given pair of boundaries within a document. BWI can be used for structured, semi-structured and free text. The patterns are token-based with special wildcards for more general rules.

CRYSTAL

CRYSTAL [Soderland, 1996] learns free-text multi-slot extraction rules named *concept definitions*, which are build of lexical, semantic and syntactic constraints. They operate on sentences or syntactic constituents that are created by a sentence analyzer. Concept definitions are induced in a bottom-up covering manner by generalizing most specific seed rules created from uncovered training instances. A seed rule is merged with the *most similar concept definition* of the initial rule base.

LP^2

This method [Ciravegna, 2003] operates on all three kinds of documents. It learns separate rules for the beginning and the end of a single slot. So called *tagging rules* insert boundary SGML tags and additionally induced *correction rules* shift misplaced tags to their correct positions in order to improve precision. The learning strategy is a bottom-up covering algorithm. It starts by creating a specific seed instance with a window of w tokens to the left and right of the target boundary and searches for the best generalization. Other linguistic NLP-features can be used in order to generalize over the flat word sequence.

RAPIER

RAPIER [Califf and Mooney, 2003] induces single slot extraction rules for semi-structured documents. The rules consist of three patterns: a pre-filler, a filler and a post-filler pattern. Each can hold several constraints on tokens and their according POS-tag- and semantic information. The algorithm uses a bottom-up compression strategy, starting with a most specific seed rule for each training instance. This initial rule base is compressed by randomly selecting

Name	Strategy	Document	Slots	Features, Auxiliaries, Characteristics
BWI	Boosting, TD	Struct, Semi	Single, Boundary	Tokens/Wildcards, AdaBoost, Voting
CRYSTAL	BU Cover	Semi, Free	Multi	Syntax, POS, Semantic, Stem
LP²	BU Cover	All	Single, Boundary	Morph, Gazetteer, POS
RAPIER	TD/BU Compr.	Semi	Single	POS, Semantic
SRV	TD Cover	Semi	Single	FOL, Relational, Semantic, Syntactic
WHISK	TD Cover	All	Multi	Syntax, POS, Semantic
WIEN	CSP	Struct	Multi, Rows	Substrings

Figure 2: Overview of the adressed methods. (TD = Top-Down, BU = Bottom-Up)

rule pairs and search for the best generalization. Considering two rules, the least general generalization (LGG) of the slot fillers are created and specialized by adding rule items to the pre- and post-filler until the new rules operate well on the training set. The best of the k rules (k-beam search) is added to the rule base and all empirically subsumed rules are removed.

SRV

SRV [Freitag, 2000] uses single-slot *first order logic* (FOL) rules to classify a given text fragment. It is an ILP system based on FOIL and is suitable for all three kinds of documents dependent on the used feature set. Rules are created in a top-down covering manner from positive and negative instances by starting with a general rule and adding literals until the rule operates well on the training set. It uses a feature-set containing simple attribute-value features and relational features.

WHISK

Another multi-slot method is WHISK [Soderland *et al.*, 1999]. It can operate on all three kinds of documents and learns single- or multi-slot rules looking similar to regular expressions. The top-down covering algorithm begins with the most general rule and specializes it by adding single rule terms until the rule makes no errors on the training set. Domain specific classes or linguistic information obtained by a syntactic analyzer can be used as additional features. The exact definition of a rule term (e.g. a token) and of a problem instance (e.g. a whole document or a single sentence) depends on the operating domain and document type.

WIEN

WIEN [Kushmerick *et al.*, 1997] is the only method listed here that operates on highly structured texts only. It induces so called *wrappers* that anchor the slots by their structured context around them. The HLRT (*head left right tail*) wrapper class for example can determine and extract several multi-slot-templates by first separating the important information block from unimportant head and tail portions and then extracting multiple data rows from table like data structures from the remaining document. Inducing a wrapper is done by solving a CSP for all possible pattern combinations from the training data.

Figure 2 gives a short overview of the adressed methods by summarizing the strategy of the algorithm, the allowed document types, the extraction output and the commonly used features, auxiliary methods or further characteristics.

3.3 The TEXTRULER System

The prototype of the TEXTRULER system was developed in [Hermann, 2009] and is currently being extended. Figure 3 shows a screenshot of the TEXTMARKER system and the integrated TEXTRULER system. Different components for the creation of labeled training documents, configuration of the selected methods and visualization of the learned rules allow the usage of the presented semi-automatic process model. Currently, prototypes of four of the six presented methods are implemented for the TEXTMARKER language and three[3] of them are rated extracting headlines for diagnoses, therapies and examinations in medical discharge letters. The following criteria adress the usefulness of the methods for a knowlegde engineer:

Comprehensibility
The comprehensibility of the learned rules is essential for the introspection, selection and further engineering of the new rules. The structure and length should not conceal the coherences between the patterns in the input document and the used features and language constructs of the rules.

Extensibility
The knowledge engineer should be able to extend and optimize the proposed rules by adapting and generalizing the language elements and their used features. A transfer of the rules to other domains and the possible improvement especially by the human way of thinking are rated.

Integratability
The learned rules need to be integrated in the existing rule set. This criteria weights the straightforwardness of the integration and transferable constructs, e.g., rules.

Usage of Features
The methods' usage of the given features and in particular the additional features of the extended rule set is of central interest for an iterative knowledge engineering. Therefore, not only the included features, but also their types and concepts for further improvements are rated.

Result, Performance
The time spent on learning and the amount of learned rules should be adequate. Finally, the extraction speed and accuracy of the rules in the focused domain are rated.

The results of the qualitative rating of the methods' current implementations are listed in figure 4. The boundary rule representation of LP^2 impairs the readability and the further engineering. The lack of integrated features and the

[3]Only LP^2, WHISK, and RAPIER have been rated, since WIEN is not applicable for the learning task.

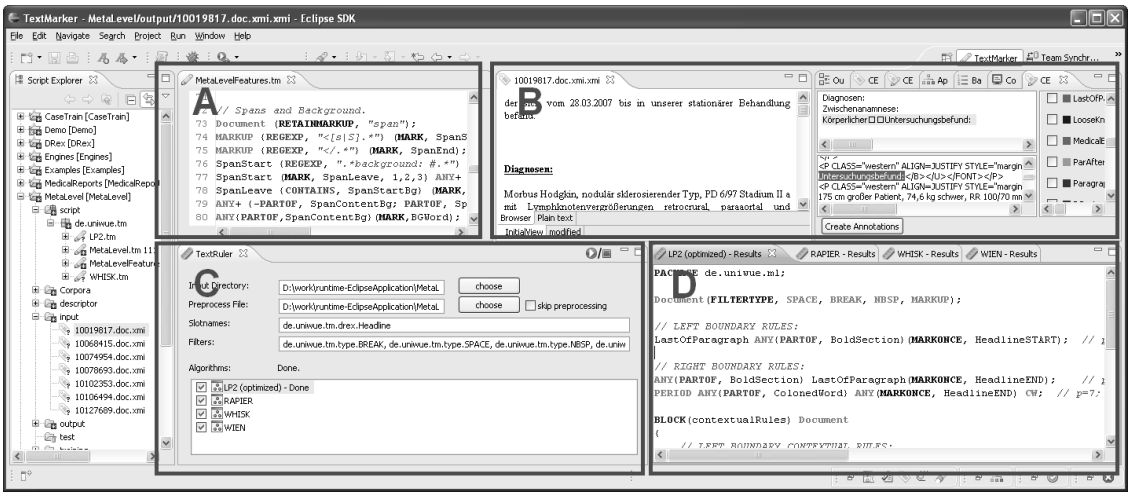

Figure 3: The TEXTRULER System: (A) Part of the TEXTMARKER development environment. (B) Editor for creating labeled training documents. (C) Control panel of the TEXTRULER system for the selection of methods and their parameters. (D) Results of the current semi-automatic development iteration.

necessary time spent on learning of RAPIER and WHISK prevent their practical usage. However, both methods are able to gain on LP^2, if their performance and features are improved. For a detailed feedback see [Hermann, 2009].

	LP^2	WHISK	RAPIER
Comprehensibility	6	8	6
Extensibility	6	7	5
Integratability	6	8	7
Usage of Features	9	2	2
Result, Performance	9	3	1
Overall	**36**	**28**	**21**

Figure 4: Qualitative rating of the current implementations in the TEXTRULER framework. (1 = weak, 10 = good)

4 Conclusions

In this paper, we have described a framework for the semi-automatic development of rule-based information extraction applications. We have presented the TEXTRULER and the TEXTMARKER systems implementing the presented approach. The TEXTRULER system provides the machine learning methods and is embedded into the TEXTMARKER system. The current prototypical implementations have been rated using a case study in the medical domain.

For future work, the implemented methods need to be improved, especially the used TextMarker language constructs and the features that are applied for annotation and information extraction. Additionally, we want to implement a more comprehensive set of learning methods covering all the discussed techniques. Furthermore, we also aim to develop novel methods that are optimized for TEXTMARKER, especially using the provided language constructs.

Acknowledgements

This work has been partially supported by the German Research Council (DFG) under grant Pu 129/8-2.

References

[Atzmueller *et al.*, 2008] Martin Atzmueller, Peter Kluegl, and Frank Puppe. Rule-Based Information Extraction for Structured Data Acquisition using TextMarker. In *Proc. of the LWA-2008 (KDML Track)*, pages 1–7, 2008.

[Califf and Mooney, 2003] Mary Elaine Califf and Raymond J. Mooney. Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction. *Journal of Machine Learning Research*, 4:177–210, 2003.

[Ciravegna, 2003] F. Ciravegna. $(LP)^2$, Rule Induction for Information Extraction Using Linguistic Constraints. Technical Report CS-03-07, Department of Computer Science, University of Sheffield, Sheffield, 2003.

[Ferrucci and Lally, 2004] David Ferrucci and Adam Lally. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Nat. Lang. Eng.*, 10(3-4):327–348, 2004.

[Freitag and Kushmerick, 2000] Dayne Freitag and Nicholas Kushmerick. Boosted Wrapper Induction. In *AAAI/IAAI*, pages 577–583, 2000.

[Freitag, 2000] Dayne Freitag. Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, 39(2):169–202, 2000.

[Hermann, 2009] Tobias Hermann. Semi-Automatic Development of Information Extraction Systems using Machine Learning Techniques (in german). Master's thesis, Univ. Wuerzburg, Dep. for Computer Science VI, 2009.

[Kushmerick *et al.*, 1997] N. Kushmerick, D. Weld, and B. Doorenbos. Wrapper Induction for Information Extraction. In *Proc. IJC Artificial Intelligence*, 1997.

[Soderland *et al.*, 1999] Stephen Soderland, Claire Cardie, and Raymond Mooney. Learning Information Extraction Rules for Semi-Structured and Free Text. In *Machine Learning*, volume 34, pages 233–272, 1999.

[Soderland, 1996] S. G. Soderland. Learning Text Analysis Rules for Domain-specific Natural Language Processing. Technical report, University of Massachusetts, Amherst, MA, USA, 1996.