Towards Learning Error-Driven Transformations for Information Extraction

Benjamin Eckstein and **Peter Kluegl** and **Frank Puppe**
University of Würzburg,
Department of Computer Science VI
Am Hubland, 97074 Würzburg, Germany
{eckstein, pkluegl, puppe}@informatik.uni-wuerzburg.de

Abstract

Transformation-based and error-driven induction of rules has proven to be a successful and applicable technique for Part-of-Speech tagging and other natural language processing tasks. This paper presents the ongoing work to create a universal algorithm for learning transformations in the domain of information extraction and semantic annotation. Such a method can be applied to different and useful tasks in real world problems, e.g., correcting systematic labeling errors of statistical models or improving data sets annotated by humans. We show the applicability of the current prototype in the domain of reference segmentation, where our method is able to reduce the labeling errors of conditional random fields.

1 Introduction

The vast availability of data and information enables many interesting applications of knowledge discovery and data mining. However, the data is often only obtainable in an unstructured form and needs to be transformed in a structured representation for further processing. Information extraction in general is one technique to conduct this task for textual documents. Here well-known types of entities and relations between these entities are identified and transfered to a structured representation like templates or relational data bases. Many methods have already been applied for these extraction tasks. One of the most popular techniques are Conditional Random Fields (CRF) and their chain structured variant linear chain CRF [Lafferty *et al.*, 2001]. CRFs model conditional probabilities with undirected graphs, are trained in a supervised fashion and discriminate label sequences that define the occurrence of entities. Although CRFs and other methods achieve remarkable results, their accuracy in real world applications is often not sufficient.

An aspect of supervised approaches is the need of labeled examples of data. Human annotators are usually required to annotate randomly sampled documents of the interesting domain with the expected outcome of the extraction task, namely the occurrences of entities and their relations. Unfortunately, this labeling task is cumbersome, cost-intensive and error-prone. Therefore, methods are required that support the maintenance of the annotated documents and enable the correction of systematic annotation errors.

This paper describes the ongoing work to create a learning algorithm for error-driven transformations that is able to solve or at least minimize the two tasks already mentioned: to improve the accuracy of statistical models like CRF and to support a human annotator to correct the labeling errors. The goal in general is to provide a universal tool to correct and adapt annotated entities for any information extraction task or semantic annotation. Error-driven transformation-based learning has been made popular in the field of natural language processing and especially for the Part-of-Speech tagging task. Brill [Brill, 1995] applied this approach to correct the Part-of-Speech information of a simple initial tagger using a induced list of transformations. The single transformations are generated based on predefined templates of possible patterns and change the tag of a word if the transformation is able to apply. We seize this idea to the broader field of generic annotations. One important point is that the induced transformations are human readable and can therefore further be adapted or extended.

The learning of extraction knowledge for information extraction is an already well studied area of research and many algorithms have been proposed, e.g., LP^2 [Ciravegna, 2003], BWI [Freitag and Kushmerick, 2000] or WHISK [Soderland *et al.*, 1999]. These methods are specialized on annotating interesting entities and use different approaches to induce suitable rules or patterns for this task. Transformations for each kind of possible correction of semantic annotations cannot be generated using a predefined template as it was done for the Brill tagger [Brill, 1995], but have to rely on a flexible way to induce matching and extraction rules. Here, the previously published approaches to rule learning for information extraction can provide valuable indications for transformation induction.

This work tries to merge the idea of error-driven transformations of semantic annotations with generic rule induction algorithms. The current implementation of the prototype is exemplarily evaluated in the domain of reference segmentation and the support of a manual annotation process is neglected in this work. We apply the algorithm on the documents labeled by a linear chain CRF, the state-of-the-art method in this domain. The results indicate that the presented prototype still requires further development, but is yet able to reduce the labeling errors.

The rest of the paper is structured as follows: Section 2 gives an overview of the background, related work and the target applications it is integrated in. The basic concepts and the algorithm for inducing transformations are described in Section 3. In Section 4, the applicability of the current prototypical implementation is investigated using a case study in the reference segmentation domain. Finally, Section 5 concludes the presented work.

2 Background

Error-driven transformation-based learning was made popular by the very successful Brill tagger [Brill, 1995]. In contrast to common approaches based on the Markov assumption, this tagger can model complex dependencies of words and tags. First, all words of a document are tagged with their most frequent tag determined by the labeled data set. In an iterated cycle, predefined triggering environments generate possible transformations that are applied on the test data set. Then, the transformation that minimizes the tagging error is selected for a transformation list. This procedure is repeated until the reduction of the tagging error is less than a given threshold. There are different means of how or when to execute the transformations in the learning process, e.g, if the selected transformation immediately processes the training instances.

There are two reasons why this algorithm cannot be directly applied for the correction of generic annotations which are focused in the presented work. On the one hand, inflexible triggering environments which define the possible shapes of rules or transformations complicate any adaption of the learning algorithm to different problems or domains. The possible combinations of matching conditions and additional constraints need to be induced by the learning algorithm itself dependent on the given training instances. This includes an arbitrary amount of features for our approach in contrast to a few predefined features for the Brill tagger. On the other hand, annotations in many domains cover more than a single token. Therefore, the frame and kind of action of a transformation also needs to be configured and selected by the learning algorithm. If the offsets of an annotation are incorrect, then the suitable transformation should result in an adjustment of its borders. However, the tagging algorithm itself only supports the determination of the label or Part-of-Speech tag of a single word. Generic annotations can admittedly be specified by an extended label set that also addresses for example the begin and end of an entity, but this approach also drastically increases the complexity of the learning algorithm.

The induction of triggering environments is one of the most important tasks of rule learning algorithms for information extraction. Their resulting extraction knowledge has to model constraints on the sequence of tokens combined with additional constraints. Since these constraints on properties or sequences are not weighted, but are applied in a deterministic fashion, it is of high importance to identify useful compositions. Many rule learning algorithms have been published and a few of them are shortly presented here.

- CRYSTAL [Soderland, 1996] tries to generalize most specific seed rules created from uncovered training examples and induces patterns in a bottom-up covering manner. The resulting extraction knowledge is able to match multiple entities and operates on sentences or syntactic constituents.
- RAPIER [Califf and Mooney, 2003] creates initial rules that cover each complete training instance and searches for best generalizations by combining two rules with the least general generalization method. Overall, rules consist of a pre-filler, a filler and a post-filler which can each hold several constraints.
- SRV [Freitag, 2000] is a system based on inductive logic programming. Rules represented by logic formulae are induced in a top-down fashion by adding additional literals.
- WHISK [Soderland *et al.*, 1999] is a top-down covering algorithm that specializes general seed rules by adding further elements and constraints. Overall, the resulting knowledge is able to identify multiple entities at once and resembles regular expressions.

Other approaches for learning extraction knowledge for information extraction rely on the idea of identifying the boundaries of the entities instead of the complete entities themselves. These methods need to provide additional constructs to assemble the interesting entities given the identified boundaries. However, due to the separation of rules to detect boundaries, the resulting extraction knowledge is more difficult to interpret by a human knowledge engineer.

- BWI [Freitag and Kushmerick, 2000] uses boosting to improve the performance of simple patterns that detect the boundaries of entities. 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.
- LP^2 [Ciravegna, 2003] is a bottom-up covering algorithm that learns rules to detect the boundaries of entities. Dynamic programming can be applied to efficiently find the best generalization. Additionally, two different kinds of rules are induced. Contextual rules identify boundaries dependent on an already extracted border. Error rules are able to shift identified boundaries and are applied to prevent unintended compositions of boundaries.

Overall, rule learning approaches for information extraction have been successfully applied in real world applications, but are recently replaced by statistical models. These black-box machine learning techniques like CRF [Lafferty *et al.*, 2001] dominate the extraction tasks due to their increased accuracy and robustness, e.g, by considering thousands of features in their inference instead of some single attributes in a modular rule. Nevertheless, these white-box rule learning approaches are very useful to point to different possibilities to induce transformations. The option to interpret, adapt and extend the resulting transformation knowledge by humans is a very important advantage in our real world applications.

Among the previously published rule learning approaches, LP^2 already provides some initial functionality for learning transformation knowledge. The error rules are able to adapt to systematic shifting of the boundaries of entities. We try to seize the general idea of LP^2 for the presented work, but neither is this approach sufficient for general transformations nor any detailed information about possible manner of implementation is available in the literature.

Nahm [Nahm, 2005] already proposed an approach for transformation-based learning to information extraction. They focus on high precision rule learner and try to improve the accuracy by inducing additional meta-rules for related entities in their local context. Overall, the applied process model resembles the approach of [Brill, 1995] and the learnt rules for increasing the recall can be compared to a more general variant of LP^2's context rules. To the best of our knowledge, only simple matching rules based on BWI are learnt that are built on already extracted entities and identify missing entities covered by their matching range. In contrast to that approach, we propose a more general transformation for all possible kind of errors that is able to improve the recall and the precision alike.

One major requirement of the proposed algorithm is the interpretable and editable result, namely the transformation rules. For this purpose, a rule language that provides elements and functionality for all possible operations and modifications has to be used for the knowledge representation of the learnt transformations.

The TEXTMARKER system [Kluegl *et al.*, 2009b] provides such a language representation that enables all kind of transformations and manipulations of given entities. TEXTMARKER is build upon UIMA [Ferrucci and Lally, 2004] and comes with a full-featured development environment that supports amongst others context sensitive auto-completion, syntax highlighting, code formatting and testing. These two advantages enable a rapid and agile prototyping of information extraction applications that are compatible to already existing components concerning the architecture. TEXTMARKER rules consist of a sequence of rule elements that match on given annotations and can be extended with quantifiers similar to regular expressions. Besides this referencing to the textual document, a single rule element contains optional conditions that have to be fulfilled if the rule matches. Additional actions of rule elements specify the behavior of the rule for example by adding or removing annotations. A detailed description of the syntax and semantic of the TEXTMARKER language can be found online[1].

The TEXTRULER [Kluegl *et al.*, 2009a] framework is an extension of the TEXTMARKER system and allows the integration of machine learning techniques in the development environment. It provides methods and classes to represent the rules, learnt by the system and further provides an interface to integrate the learning algorithms directly into the graphical user interface. Thus the user can directly choose between different preprocessing scripts, learning corpora and parameter settings, if the learning technique allows them. After running the learning method, the user can directly access and alter the learnt rules for further usage. This way, TEXTRULER creates a connection between the manual creation of rules and the learning process by machine, enabling it to exploit the advantages of both approaches. TEXTRULER provides up to now prototypical implementations of RAPIER, WHISK, LP^2 and other approaches.

These systems build a very suitable technical basis for the implementation of the presented transformation-based learning algorithm. The TEXTMARKER system and its TEXTRULER extension are both open source and freely available for download[2].

3 Method

In this Section, some ideas of transformation-based tagging were adopted and extended to develop the concept of transformation-based learning of rules. Analogously to the Brill tagger, the structure of the transformation-based learning algorithm can be broken down into two basic steps. Those are the preprocessing of the unstructured documents by an initial state annotator and the repetitive part of learning and selecting the rules. This first step is considered to be already performed, e.g., an arbitrary statistical model labeled the documents or a human annotated the documents for creating a gold standard. The second part can be further split into two components: The generation of a set of potential rules and the evaluation and selection of those that will be added to the transformation list. Depending on the implementation, these steps can be progressed simultaneously, since however those components play an important role, they will be discussed separately in the following. One important detail that stands in contrast to transformation-based tagging consists in the fact that there is no predefined list of triggering environments which is used to generate a list of transformations. A separate list of possible rules is created for each iteration of the algorithm based on discovered differences between the two sets of documents: the labeled gold standard and its variant labeled by the initial state annotator. This is essential as the big variety of domains that have to be covered cause many possible constellations of dependencies between words, syntax and annotations. While a part-of-speech tagger works in one certain domain, the number of usable rules is limited whereas a rule learning algorithm should be able to adapt to almost every possible domain. Another difference is the behavior of the transformation that is not only changing the label of a token, but has to manipulate the entities in different manners. However, similar to the Brill tagger, the induced rules of the presented approach are executed in an imperative fashion in the order they are selected for the transformation list.

A typical example for a common error committed by statistical learning methods during the analysis of scientific references is the annotation of editors erroneously as authors. The only notable difference is the position in the text or the usage of the keyword "editors" or "eds.". In order to create a correction rule, a transformation-based learning algorithm needs to induce a rule that changes the author entity to an editor entity if it starts or ends with such a keyword. The segmentation of references is a well-known domain for the evaluation of information extraction techniques and is therefore also used in the further examples.

3.1 Error Types

The labeled data annotated by the initial state annotator and the corresponding gold standard are the basis of the rule learning process. A transformation rule basically consists of two parts: the trigger and the transformation. To build a potential rule set, the differences between training data and gold standard are determined. These differences will from now on be called error. In order to identify these errors, the training data is compared to the gold standard: by filtering out the correctly labeled annotations, the two received sets contain the erroneous annotations as the starting points and the correct annotations as the targets of the transformations. These two lists are divided into pairs to create the sought errors. From now on, we will differentiate between four basic types of errors, which represent the required transformation types for future rules:

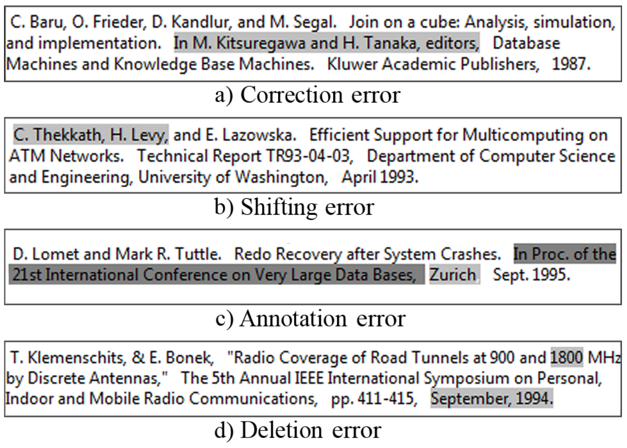
- **Correction** - an element has been assigned to the wrong annotation type. The correct type has to be determined to reassign it.

  Example: if the editors of a scientific reference have been annotated as authors, then the dependencies can easily be determined by looking for the keyword "editors" (see Figure 1a).

- **Shifting** - the borders of an annotation are wrong, so either some associated elements are outside the bor-

[1]http://tmwiki.informatik.uni-wuerzburg.de/

[2]http://sourceforge.net/projects/textmarker/


C. Baru, O. Frieder, D. Kandlur, and M. Segal. Join on a cube: Analysis, simulation, and implementation. In M. Kitsuregawa and H. Tanaka, editors, Database Machines and Knowledge Base Machines. Kluwer Academic Publishers, 1987.

a) Correction error

C. Thekkath, H. Levy, and E. Lazowska. Efficient Support for Multicomputing on ATM Networks. Technical Report TR93-04-03, Department of Computer Science and Engineering, University of Washington, April 1993.

b) Shifting error

D. Lomet and Mark R. Tuttle. Redo Recovery after System Crashes. In Proc. of the 21st International Conference on Very Large Data Bases, Zurich Sept. 1995.

c) Annotation error

T. Klemenschits, & E. Bonek, "Radio Coverage of Road Tunnels at 900 and 1800 MHz by Discrete Antennas," The 5th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 411-415, September, 1994.

d) Deletion error

Figure 1: The basic error types.

ders or some covered elements are falsely part of the annotation. The borders must be moved.

Example: the name of the last author in the list has not been associated with the annotation before. Its borders have to be expanded (see Figure 1b).

- **Annotation** - an annotation is missing completely. It has either not been covered by the preprocessing steps or its annotation has been removed by a formerly executed rule, e.g., a shifting rule. This corresponds with the general rule learning task of other learning algorithms.

  Example: A location has been marked as part of a booktitle by the preprocessing script. After shifting the borders of the booktitle, the location isn't annotated any more. It has to be reassigned to its correct type (see Figure 1c).

- **Deletion** - one or more tokens have been falsely assigned to an annotation and have to be deleted.

  Example: Some number in a title resembles a date and has been annotated. It has to be deleted as there is no other annotation type that fits its properties (see Figure 1d).

By assigning each error to one of those error types, we can determine the basic behavior of the corresponding transformation rule, e.g., a correction-rule has to delete the present annotation and assign a new one to all covered elements, while a shifting rule has to determine the borders of the present and the aspired annotation before adjusting them. This way we get a list of transformations, that must be applied to our training documents.

3.2 Rule Induction

Each transformation can serve as a seed example for the rule induction. In order to only change the desired annotations, we have to add some conditions to these rules. This means, we have to find the characteristics, that differentiate the affected annotations from others. The more conditions we add to the trigger, the more specific the rule will be. One essential condition are the rule elements, covered by the annotation, which can also be annotations and entities themselves. As one token can be part of different annotations, there is usually more than one different sequence of rule elements, that represent the same text block. Each of these sequences is a potential candidate for our final rule while there can be big differences in their matching characteristics in our corpora. As we cannot rate those rules, before having built them, we have to either choose some of them according to fixed requirements or build every possible rule, what can result in a huge list of final rules. After analyzing the received rules, we discovered that a simple sequence of tokens almost always resulted in an underfitting problem, e.g., a booktitle consists of a large series of different word types, which could not be fit in a pattern that resembled those of other booktitles. So we decided to only consider the rule elements before and/or after the borders of an annotation similar to boundary detectors like LP^2 [Ciravegna, 2003]. However those detectors are integrated in a single transformation in order to preserve the context of the transformation. This way, most of the basic rules have better matching characteristics, but are too general for our purpose. So we need to find other conditions to specialize them. This is achieved in a similar manner as done by WHISK [Soderland *et al.*, 1999]. Here are some additional conditions we used in our work:

- **StartsWith / EndsWith** - the first/last token of the examined annotation is a specific keyword or an instance of a certain annotation.

  Example: the page numbers of a book usually start with the keyword "pages".

- **Contains** - the specified rule element exists between the borders of the examined annotation, so it is a part of it.

  Example: an element, containing the name of a month is very likely a date.

- **Before / After** - the annotation is located directly before/behind a specified annotation or keyword.

  Example: the title of a publication is usually located behind the name of the author.

Again, there can be more than one instance of each condition type, as the affected token can be part of different annotations, each having its own matching characteristics. So after determining the possible conditions, we get two lists for every error example in our learning documents. One containing the conditions and the other the basic rules, created before. By combining elements from both lists, we receive our final set of rules. Note, that a rule can contain multiple conditions or none at all, so the rule list can grow large, when allowing for all possible combinations. To limit the size of the rule lists, we decided to only allow one instance of the same condition type for each rule. This set of rules is now being rated to create our output set for this iteration.

3.3 Rule Selection

The rating of each rule follows a simple scheme. Each rule is separately applied on the preprocessed training data set. Then, these documents are compared with the gold standard and the coverage of fixed and caused errors is determined. The best rated transformation rule is then added to the final transformation list. It is important to decide, which rating is to prefer: some rules could fix many errors but in return cause others which are much more crucial. On the other side, rules that correct few errors without causing new ones, could be ineffective, when being applied to documents outside the training corpus. It must be carefully considered which is the best way without causing under- or overfitting, as the choice has a major impact on the outcome. For the current implementation of the algorithm, a simple heuristic was applied that weights the ratio of corrected errors to newly introduced errors. The chosen rule is applied to the learning corpus, thus affecting the rating of all other rules in the list. By using the covering statistics of all rules, we can now once more choose the best rule and add it to the output list. This can be performed without further testing of the rules on the training data set since already corrected examples are removed from the weighting of the rules. Therefore only rules are further selected for the transformation list that correct remaining errors. This procedure is repeated until no useful rules can be found. The number of allowed rules per iteration can further be limited to increase the performance of the system or to affect its way of working. Now, the algorithm can either jump into the next iteration and determine new errors in the learning corpus, modified by the formerly generated rules, or it will terminate and return the list of learnt transformation rules. The choice of three factors greatly affect the output of the learning algorithm. Those are the number of iterations, the maximum number of created rules per error and the allowed complexity of combinations between different conditions per rule.

The algorithm for learning transformations is currently still under further development and will be released as part of the TEXTMARKER system in future.

4 Indicative Evaluation

The proposed algorithm is evaluated in the domain of reference segmentation. The data set is composed of freely available corpora and only features are applied commonly used in this domain. To be more precisely, we used the base line model and the first fold of the experimental study in Toepfer et al. [Toepfer *et al.*, 2010]. The training data set contains 315 references and the test data set consists of 137 references. An implementation of the linear chain CRF[3] served as an initial state annotator, so the purpose of the learning algorithm was to correct and complete its work. The following annotations have been considered: *author, booktitle, editor, institution, journal, location, note, pages, publisher, tech, title* and *volume*. Two different parameter settings are applied on the same data sets (cf. Table 1). We used the common token-based evaluation measures precision, recall, F_1 and $F_1\varnothing$ (an average over the F_1 scores of each type of entity). With true positives (tp), false positives (fp) and false negatives (fn) we receive the following formulae:

$$precision = \frac{tp}{tp + fp}, \quad recall = \frac{tp}{tp + fn},$$

$$F_1 = \frac{2 * precision * recall}{precision + recall}.$$

[3]The implementation of Mallet was used: http://mallet.cs.umass.edu/

Table 1: Parameter settings for the evaluation

test run	iterations	rules / error	cond / rule
1	7	50	6
2	20	20	1

Table 2: First run on training data

	prec	recall	F_1	$F_1\varnothing$
before	0.957	0.957	0.957	0.826
after	0.975	0.975	0.975	0.919

Table 3: First run on test data

	prec	recall	F_1	$F_1\varnothing$
before	0.953	0.953	0.953	0.783
after	0.918	0.950	0.934	0.816

Table 4: Second run on training data

	prec	recall	F_1	$F_1\varnothing$
before	0.957	0.957	0.957	0.826
after	0.970	0.973	0.972	0.919

Table 5: Second run on test data

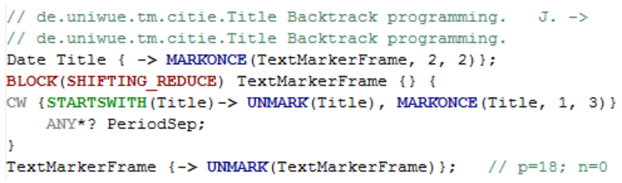
	prec	recall	F_1	$F_1\varnothing$
before	0.953	0.953	0.953	0.783
after	0.958	0.960	0.959	0.854

4.1 Results

During the test runs, each learnt set of rules has been applied on both the training and the test corpus to determine the number of covered examples and therefore the quality of the created rules. The algorithm was able to learn rules for each of the four error types, while their quality varied widely, but not every example could be covered. The parameters of the first test run featured a large number of rules containing many different conditions for each example. When applied on the learning corpus, they achieved good results as all measured values could be improved (see Table 2). But when being applied on the test corpus, the same rules did not achieve the results, we hoped for. Table 3 shows, that all measured values deteriorated after applying the rules except for the average F_1 score. This indicates that some rules did a good job while some others totally messed up the results. For the second test run we chose a higher number of iterations and a smaller number of rules with only one added condition per example. These more basic rules could improve the values of the learning corpus nearly as good as those of the first test run (see Table 4). This is not very surprising, as the rules were applied on the same examples they were learnt from, but it shows their functionality. Then again, we used the rules on the test corpus and measured the received results. Table 5 shows, that all values could be improved this time.

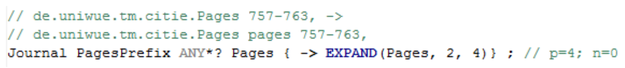
4.2 Discussion

The form and quality of the chosen rules varied a lot. Below we will have a closer look at some of those rules to show the strengths and weaknesses of the learning algorithm. Figure 2 shows a shifting rule, which reduces the borders of the annotation *title* if necessary. The occurrence of *Date* indicates, that the examined annotation is located behind a date and the control block, containing the rule elements *CW ANY*? PeriodSep* shows, that a *title* ends with the first occurrence of a period. The commented part on top shows the covered text of the example, on which the rule was learnt while the values p and n below show the positive or negative corrections on the training data during testing. This rule successfully corrected some errors and therefore had a good rating. The rule shown by Figure 3 is



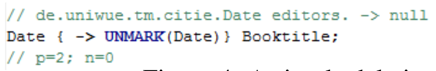
```
// de.uniwue.tm.citie.Title Backtrack programming.   J. ->
// de.uniwue.tm.citie.Title Backtrack programming.
Date Title { -> MARKONCE(TextMarkerFrame, 2, 2)};
BLOCK(SHIFTING_REDUCE) TextMarkerFrame {} {
CW {STARTSWITH(Title)-> UNMARK(Title), MARKONCE(Title, 1, 3)}
    ANY*? PeriodSep;
}
TextMarkerFrame {-> UNMARK(TextMarkerFrame)};   // p=18; n=0
```

Figure 2: A good example for a reduction rule.



```
// de.uniwue.tm.citie.Pages 757-763, ->
// de.uniwue.tm.citie.Pages pages 757-763,
Journal PagesPrefix ANY*? Pages { -> EXPAND(Pages, 2, 4)} ; // p=4; n=0
```

Figure 3: An expansion (shifting) rule.



```
// de.uniwue.tm.citie.Date editors. -> null
Date { -> UNMARK(Date)} Booktitle;
// p=2; n=0
```

Figure 4: A simple deletion rule.

an expansion rule (a subtype of shifting rules) and works on the annotation type *Pages*. If the algorithm discovers an instance of the type *pagesPrefix* preceding the examined rule element, it expands the borders of the annotation to include it. The additional condition *Journal* ensures, that only tokens directly after this annotation are examined and no earlier or later occurring instances of *Pages* are affected. As a final example, consider the simple but efficient rule shown in Figure 4. As in the used references a date never precedes a booktitle, the rule simply deletes every annotation *date* that does.

Although the rules learnt in the first test run did not improve the test corpus, the algorithm generated some good rules, which could be developed further. The learnt transformations of the second run worked successfully and confirmed the possibilities of the shown method. The tests showed, that overfitting is the biggest problem, when applying the rules, but other factors like the small training corpus and the limits of the TextMarker language also affected these results. We nevertheless received good results and a set of rules which can be further used and improved.

5 Conclusions

This paper presented the ongoing work to create an error-driven transformation-based rule learning method for the correction of arbitrary annotation errors. The current prototype is based on the idea of transformation-based learning in general merged with the rule induction of well-known rule learning algorithms for information extraction. Early results indicate the usefulness of the presented approach, but also point out the flaws, like the tendency for overfitting and strong dependencies on the selected parameters.

The next steps for the development involve a further increase of the readability of the learnt transformations by generating neatly arranged rules and by extending the TEXTMARKER language for representing transformations more straightforwardly. Besides that, there remain many means to speed up the algorithm, e.g., using heuristics to prune the list of generated rules that need to be tested on the documents. The most important improvement will be the reduction of overfitting. This can be achieved by dividing the training data set during the learning of transformations. Similar to a cross fold evaluation, the training data set needs to be split into disjoint partitions. Each partition is then used to learn rules that are however tested on the left out partition during the learning step.

Acknowledgements

We like to thank the reviewers for their helpful and constructive critique. Unfortunately, not all of their comments could be included in this paper, but they will improve our further research in this domain.

References

[Brill, 1995] Eric Brill. Transformation-based Error-driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Comput. Linguist.*, 21(4):543–565, 1995.

[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.

[Kluegl *et al.*, 2009a] Peter Kluegl, Martin Atzmueller, Tobias Hermann, and Frank Puppe. A framework for semi-automatic development of rule-based information extraction applications. In Melanie Hartmann und Frederik Janssen, editor, *Proc. LWA KDML Workshop*, pages 56–59, 2009.

[Kluegl *et al.*, 2009b] Peter Kluegl, Martin Atzmueller, and Frank Puppe. TextMarker: A Tool for Rule-Based Information Extraction. In Christian Chiarcos, Richard Eckart de Castilho, and Manfred Stede, editors, *Proceedings of the Biennial GSCL Conference 2009, 2nd UIMA@GSCL Workshop*, pages 233–240. Gunter Narr Verlag, 2009.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proc. 18th ICML*, pages 282–289, 2001.

[Nahm, 2005] Un Yong Nahm. Transformation-Based Information Extraction Using Learned Meta-rules. In *CICLing*, pages 535–538, 2005.

[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.

[Toepfer *et al.*, 2010] Martin Toepfer, Peter Kluegl, Andreas Hotho, and Frank Puppe. Conditional random fields for local adaptive reference extraction. In *Proc. LWA KDML Workshop*, 2010.