

Learning Approaches to Wrapper Induction

Gunter Grieser
TU Darmstadt
Fachbereich Informatik
Alexanderstraße 10
64283 Darmstadt, Germany
grieser@informatik.tu-darmstadt.de

Steffen Lange
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
lange@dfki.de

Abstract

The number, the size, and the dynamics of Internet information sources bears abundant evidence of the need of automation in information extraction (IE). This paper deals with the question of how such extraction mechanisms can automatically be created by invoking learning techniques.

The underlying scenario of system-supported IE is putting certain constraints on the available training examples. Therefore, the traditional approaches to formal language learning do not capture the kind of problems to be solved when learning the corresponding extraction mechanisms.

We illustrate the resulting differences by studying the problem of learning a particular type of extraction mechanisms (so-called island wrappers). We show how to decompose this learning problem into different sub-problems that can be handled independently and in parallel. Moreover, we relate the learning problems on hand to the problems that learning theory papers originally address and point out what they have in common and where the differences are.

Motivation and Introduction

The work reported in the present paper mainly draws its motivation from ongoing research related to knowledge discovery and information extraction (IE) in the World Wide Web. Documents prepared for the Internet in HTML, in XML or in any other syntax have to be interpreted by browsers sitting anywhere in the World Wide Web. For this purpose, the documents do need to contain syntactic expressions which are controlling its interpretation including its visual appearance and its interactive behaviour. These syntactic expressions are usually hidden from the user and obviously apart from the user's interest. The user is typically interested in the information itself. Accordingly, the user deals exclusively with the desired contents, and a system for IE should deal with the syntax.

In a characteristic scenario of system-supported IE, the user is taking a source document and is highlighting representative pieces of information that are of interest. It is left to the system to understand how the target information is

Copyright © 2001, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

wrapped into syntactic expressions and to learn a procedure (henceforth called wrapper) that allows for an extraction of this information.

For illustration, see the web page in Figure 1 which concerns details about the program committee of the special track Knowledge Discovery and Data Mining of this year's Florida AI Symposium.

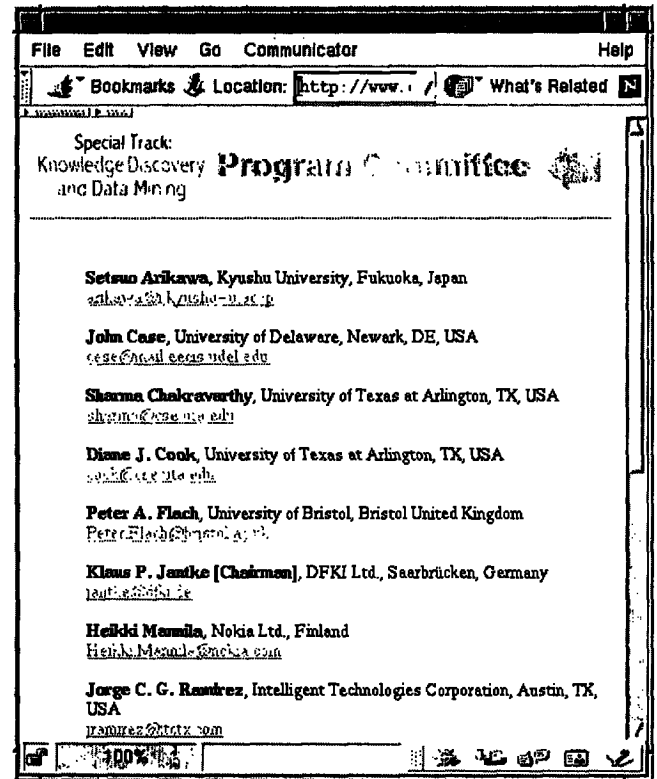


Figure 1: Visual appearance of a sample document D

In our example, a potential user is interested in the names and corresponding e-mail addresses of the pc-members.

Figure 2 displays the HTML source of document D. Special text segments occur at the beginning and the end of the information the user is interested in. In this example, the names of the pc-members are surrounded by the strings '' and '' resp. '_[Chairman]()', while the

There are several ways to present information about formal languages to be learned. The basic approaches are defined via the concepts *text* and *informant*, respectively. A text is just any sequence of words exhausting the target language. An informant is any sequence of words labelled alternatively either by 1 or 0 such that all the words labelled by 1 form a text for the target language L , whereas the remaining words labelled by 0 constitute a text of the complement of L . In all what follow, we exclusively deal with the case of learning from text.

An algorithmic learner (henceforth, called IIM) receives as inputs larger and larger initial segments of a text t for a target language L and generates as its outputs hypotheses. An IIM learns a target language L from text t , if the sequence of its outputs stabilizes on a hypothesis that correctly describes L . Now, an IIM is said to learn L from text, if it learns L from every text for it. Furthermore, some language class \mathcal{C} is said to be learnable from text, if there is an IIM which learns every language $L \in \mathcal{C}$ from text. By *LimTxt* we denote the collection of all indexable classes \mathcal{C} for which there is an IIM that learns \mathcal{C} from text.

The collection of all *LimTxt*-identifiable classes can be characterized in terms of finite ‘tell-tale sets’.

Theorem 1 (Angluin 1980) *Let \mathcal{C} be an indexable class. $\mathcal{C} \in \text{LimTxt}$ iff there are an indexing $(L_j)_{j \in \mathbb{N}}$ of \mathcal{C} and a procedure p that, given any $j \in \mathbb{N}$, enumerates a finite set S_j such that (i) and (ii) are fulfilled, where*

- (i) for all $j \in \mathbb{N}$, $S_j \subseteq L_j$.
- (ii) for all $j, k \in \mathbb{N}$, if $S_j \subseteq L_k$ then $L_k \not\subseteq L_j$.

Island Wrappers

Next, we define a particular type of wrappers, so-called island wrappers (cf. (Thomas 1999a; 1999b)), that describe a particular way of how interesting information may be wrapped into an HTML document. This approach rests upon the following assumptions.

- (i) Information of the same type visually appears in the same style in a document and, vice versa, the visual appearance allows for grouping information.
- (ii) The visual appearance of information has to be encoded in a document, and therefore the corresponding text parts are surrounded by HTML commands that in turn may help to find these text parts in the document.

More specifically, it is assumed that some corresponding text part v can be identified by particular strings l and r that occur in the HTML document to the left and to the right of v , respectively. The surrounding strings l and r , which are by no means limited to contain only formatting commands, are called *delimiters*. The text part lvr is said to be an *island*.

Island wrappers generalize this basic approach by specifying of how interesting n -tuples $\langle v_1, \dots, v_n \rangle$ may be wrapped into HTML documents. Here, the corresponding left and right delimiters are not uniquely fixed. In contrast, they may belong to possibly infinite languages of admissible delimiters (so-called *delimiter languages*). As we see below, a particular island wrapper is uniquely characterized by its delimiter languages.

In general, given any document $d \in \Sigma^+$, an island wrapper W may be understood as a finite description of a mapping that assigns a finite set $S_W(d)$ of n -tuples to d , where $S_W(d)$ contains all and only the n -tuples that are wrapped into d , accordingly.

More formally, let $n \geq 1$, let $L_1, R_1, \dots, L_n, R_n$ be delimiter languages, and let $W = (L_1, R_1, \dots, L_n, R_n)$ be the corresponding island wrapper. Then, the island wrapper W defines the following mapping S_W from documents to n -ary relations: Given any document d , we let $S_W(d)$ be the set of all n -tuples $\langle v_1, \dots, v_n \rangle \in (\Sigma^+)^n$ for which there are $x_0 \in \Sigma^+, \dots, x_n \in \Sigma^+, l_1 \in L_1, \dots, l_n \in L_n$ and $r_1 \in R_1, \dots, r_n \in R_n$ such that conditions (i) to (iii) are fulfilled, where

- (i) $d = x_0 l_1 v_1 r_1 \dots l_n v_n r_n x_n$.
- (ii) for all $i \in \{1, \dots, n\}$, v_i does not contain a substring belonging to R_i , i.e., $v_i \in \tilde{\Sigma}_{R_i}^+$.
- (iii) for all $i \in \{1, \dots, n-1\}$, x_i does not contain a substring belonging to L_{i+1} , i.e., $x_i \in \tilde{\Sigma}_{L_{i+1}}^+$.

Conditions (ii) and (iii) imply that the extracted strings are as short as possible and that the distance between them is as small as possible. This, in particular, helps to avoid that extracted strings, which are supposed to belong to different n -tuples, are grouped in the same n -tuple.

Learning Island Wrappers

In this section, we deal with the problem of how to learn wrappers from examples. In the long-term, we are interested in learning devices that learn extraction mechanisms implementing the mappings defined by the corresponding wrappers. The latter is one of the central topics of a joint research and development project named LEXIKON on information extraction from the Internet. This project is supported by the German Federal Ministry for Economics and Technology. Beside the authors’ institutions, the project consortium comprises other academic research teams at the University Leipzig, the University of Koblenz-Landau, and the Bayerische Hypo- und Vereinsbank AG, respectively, and the two software houses rzw_cimdata and Dr. Stephan & Partner.

Since an island wrapper is completely determined by its delimiter languages, the overall learning problem reduces to the problem of finding these delimiter languages. On the first glance, this seems to be a particular instance of the traditional learning problems described above. However, a potential user does not directly provide elements of the delimiter languages as example to the IE system. Instead, the user marks interesting text parts in the HTML documents under inspection, and thus the IE system receives only implicit information about the delimiter languages to be learned. Consequently, the approaches developed for traditional language learning do only translate indirectly – after an appropriate adaptation – to the setting of language learning from *marked text*.

The relevant details are as follows.

Now, suppose that a user marks an interesting n -tuple $\langle v_1, \dots, v_n \rangle$ in a document d under inspection.

Herewith, the user simultaneously marks the corresponding starting and end positions of these strings. Moreover, assuming that each v_i ends before v_{i+1} starts, the user samples the document into $2n + 1$ consecutive text parts $u_0, v_1, u_1, \dots, v_n, u_n$. Hence, the string $u_0 v_1 u_1 \dots v_n u_n$ equals d . Finally, such a $2n+1$ -tuple $\langle u_0, v_1, u_1, \dots, v_n, u_n \rangle$ is said to be an n -marked document.

Let $W = (L_1, R_1, \dots, L_n, R_n)$ be an island wrapper and let $m = \langle u_0, v_1, u_1, \dots, v_n, u_n \rangle$ be an n -marked document. Then, m is said to be an *example* for W if conditions (i) to (iii) are fulfilled, where

(i) $u_0 \in \Sigma^* \circ L_1$ and $u_n \in R_n \circ \Sigma^*$.

(ii) for all $i \in \{1, \dots, n\}$, $v_i \in \tilde{\Sigma}_{R_i}^+$.

(iii) for all $i \in \{1, \dots, n-1\}$, $u_i \in R_i \circ \tilde{\Sigma}_{L_{i+1}}^+ \circ L_{i+1}$.

Now, an n -marked text t for an island wrapper W is an infinite sequence of n -marked documents that serve as examples for W . Similarly as above, it is required that t exhausts the set of all possible examples for W , i.e., every n -marked document that constitutes an example for W must eventually appear.

Finally, a wrapper learner (henceforth called WIM) receives as input larger and larger initial segments of a marked text for some target wrapper W and generates as outputs hypotheses. Learning took place iff the sequence of hypotheses stabilizes on some description of a wrapper W' such that $S_W(d) = S_{W'}(d)$ for all documents $d \in \Sigma^+$.

Collection of relevant learning problems

Conceptually, when learning an island wrapper from marked text, one may proceed as follows: In a first step, the overall learning problem has to be decomposed into several individual learning problems. In a second step, learning algorithms have to be invoked to solve the derived individual learning problems independently and in parallel. In a concluding step, the solutions of the individual problems have to be used to determine the defining delimiter languages and to fix the hypothesized island wrapper.

In general, there are three types of individual learning problems that have to be solved.

So, let $W = (L_1, R_1, \dots, L_n, R_n)$ be the target island wrapper and let $\langle (u_0, v_1, u_1, \dots, v_n, u_n) \rangle_{j \in \mathbb{N}}$ be the n -marked text presented.

First, the subsequence $(u_0)_j \in \mathbb{N}$ provides information about the left delimiter language L_1 , since every u_0 has a suffix belonging to L_1 . However, from the IE system's perspective, it is by no means clear where the corresponding suffix starts in u_0 . But it is clear that $(u_0)_j \in \mathbb{N}$ forms a text for a language of type 1:

• $T_1(L) = \Sigma^* \circ L$.

Second, the subsequence $(u_n)_j \in \mathbb{N}$ provides information about the right delimiter language R_n , since every u_n has a prefix belonging to R_n . But the n -marked text contains some more information concerning R_n . By definition, the strings v_n marked by the user do not contain any substring that belongs to R_n . To formalize this, let $\#$ be any symbol not occurring in Σ . Then, the sequence $(v_n \# u_n)_j \in \mathbb{N}$ constitutes a text for a language of type 2:

• $T_2(L) = \tilde{\Sigma}_L^+ \circ \{\#\} \circ L \circ \Sigma^*$.

Third, the subsequence $(u_1)_j \in \mathbb{N}$, for instance, simultaneously provides information about the right and the left delimiter language R_1 and L_2 , respectively, since u_1 has a prefix and a suffix belonging to R_1 and L_2 , respectively. As above, by definition, the n -marked text contains some more information concerning both delimiter languages. Putting everything together, one directly sees that the sequence $(v_1 \# u_1)_j \in \mathbb{N}$ forms a text for a language of type 3:

• $T_3(L, L') = \tilde{\Sigma}_L^+ \circ \{\#\} \circ L \circ \tilde{\Sigma}_{L'}^* \circ L'$.

Hence, by exploiting all the information which an n -marked text contains, one comes up with $n + 1$ different traditional problems of learning languages from ordinary text. More formally, the following learning problems are of interest.

Definition 1 Let \mathcal{C} be an indexable language class. For all $i \in \{0, \dots, 3\}$, the learning problem $LP_i(\mathcal{C})$ can be solved iff $T_i(\mathcal{C}) \in \text{LimTxt}$, where $T_0(\mathcal{C}) = \mathcal{C}$, $T_1(\mathcal{C}) = \{T_1(L) \mid L \in \mathcal{C}\}$, $T_2(\mathcal{C}) = \{T_2(L) \mid L \in \mathcal{C}\}$, and $T_3(\mathcal{C}) = \{T_3(L, L') \mid L, L' \in \mathcal{C}\}$.

Here, $LP_0(\mathcal{C})$ serves as kind of master problem when relating the resulting learning problems $LP_1(\mathcal{C})$ till $LP_3(\mathcal{C})$ to one another. We are interested in answering questions like the following: Is it possible to solve the learning problems $LP_1(\mathcal{C})$, $LP_2(\mathcal{C})$ or $LP_3(\mathcal{C})$ provided one knows that there is a solution for the problem $LP_0(\mathcal{C})$? Questions of this kind will intensively be studied in the next subsection.

Relations between the relevant learning problems

We start our discussion with an illustrating example. Let $\Sigma = \{a, b, c\}$. For all $n \in \mathbb{N}$, let $L_0 = \{a^m b \mid m \geq 1\} \cup \{c\}$ and $L_{n+1} = \{a^m b \mid 1 \leq m \leq n+1\} \cup \{c, ca\}$. Moreover, we let \mathcal{C}_A be the collection of all these languages.

First, we consider the learning problem $LP_1(\mathcal{C}_A)$. An appropriate IIM M may work as follows.

IIM M : On input w_0, \dots, w_m , check whether some of the strings w_0, \dots, w_m ends with a . If no such string occurs, output a description for $\Sigma^* \circ L_0$. Otherwise, return a description for $\Sigma^* \circ L_1$.

Obviously, M witnesses $T_1(\mathcal{C}_A) \in \text{LimTxt}$. Does this insight help to derive an answer to the question of whether or not the learning problem $LP_2(\mathcal{C}_A)$ can be solved? The ultimate answer is no, since it turns out that $T_2(\mathcal{C}_A) \notin \text{LimTxt}$.

To see this, assume the contrary, i.e., let M be an IIM that learns $T_2(\mathcal{C}_A)$ in the limit from text. First, it is easy to see that $\tilde{\Sigma}_{L_i}^+ = \tilde{\Sigma}_{L_j}^+$ for any $i, j \in \mathbb{N}$. Based on M , one can easily define an IIM M' that LimTxt -identifies the indexable class $\{L \circ \Sigma^* \mid L \in \mathcal{C}_A\}$. Next, by Theorem 1, there is a finite set $S_0 \subseteq L_0 \circ \Sigma^*$ such that $S_0 \subseteq L \circ \Sigma^*$ implies $L \circ \Sigma^* \not\subseteq L_0 \circ \Sigma^*$, for any $L \in \mathcal{C}_A$. For the ease of argumentation, assume that some string in S_0 has a prefix of form $a^{n'} b$. Let n be the maximal index n' . Clearly, $L_n \circ \Sigma^* \subset L_0 \circ \Sigma^*$. On the other hand, one directly sees that $S_0 \subseteq L_n \circ \Sigma^*$. But this contradicts our assumptions that S_0 serves as a finite tell-tale set for L_0 , and thus $T_2(\mathcal{C}_A) \notin \text{LimTxt}$.

To sum up the discussion of our illustrating example: Knowing that the learning problem $LP_1(C)$ can be solved for some indexable class C does not imply that one can solve the learning problem $LP_2(C)$ as well.

Surprisingly, this insight generalizes as follows:

Theorem 2 *Let $i, j \in \{1, \dots, 4\}$ with $i \neq j$. Then, there is an indexable class C such that assertions (i) and (ii) are fulfilled, where*

- (i) *it is possible to solve problem $LP_i(C)$.*
- (ii) *it is impossible to solve problem $LP_j(C)$.*

Proof. For each relevant pair i, j , we have to provide an indexable class that meet (i) and (ii). Due to space constraints, we only sketch some simple cases.

As argued above, the learning problems $LP_0(C_A)$ and $LP_1(C_A)$ can be solved. A careful analysis shows that the same is true for $LP_3(C_A)$, while $LP_2(C_A)$ cannot be solved.

Now, let $\Sigma = \{a, b\}$ and let C_B be the collection of the following languages L_n , where, for all $n \in \mathbb{N}$, $L_0 = \{ab^m a \mid m \geq 1\}$ and $L_{n+1} = L_0 \setminus \{ab^{n+1} a\}$. As it turns out, the learning problems $LP_0(C_B)$ and $LP_1(C_B)$ cannot be solved. In contrast, the learning problems $LP_2(C_B)$ and $LP_3(C_B)$ have a solution. This is mainly due to the fact that, for all $n \in \mathbb{N}$, $\tilde{\Sigma}_{L_{n+1}}^+$ contains exactly one string that belongs to L_0 , namely the string $ab^{n+1} a$. This allows one to distinguish the languages $T_2(L_0)$ and $T_2(L_{n+1})$ as well as $T_3(L_0)$ and $T_3(L_{n+1})$, respectively. \square

Consequently, there are indexable classes C such that

- (i) knowing that there is a solution for one of the learning problems does not help to solve the other ones and, vice versa,
- (ii) knowing that some learning problem cannot be solved does not mean that one cannot solve the other ones.

After showing that all the learning problems defined are somehow independent, further investigations deal with the question under what circumstances learning becomes possible. To answer this question, we provide a kind of case study concerning learning problem $LP_2(C)$.

One particular learning problem

Wright (1989, see also (Motoki, Shinohara, & Wright 1991)) introduced the following sufficient condition for limit learning from text.

Definition 2 (Wright 1989) *Let C be an indexable class. C has infinite elasticity, if there are an infinite sequence $(w_j)_{j \in \mathbb{N}}$ of strings from Σ^* and an infinite sequence of languages $(L_j)_{j \in \mathbb{N}}$ of languages in C such that, for all $k \geq 1$, $\{w_0, \dots, w_{k-1}\} \subseteq L_k$ and $w_k \notin L_k$.*

C has finite elasticity if it does not have infinite elasticity.

Theorem 3 (Wright 1989) *Let C be an indexable class that has finite elasticity. Then, $C \in \text{LimTtxt}$.*

Assume that a given indexable class C has finite elasticity. By Theorem 3, we know that $C \in \text{LimTtxt}$. Does this imply $T_2(C) \in \text{LimTtxt}$, as well? If we can show that $T_2(C)$ has finite elasticity as well, then we are immediately done. However, this is impossible.

Theorem 4 *There is an indexable class C such that conditions (i) and (ii) are fulfilled, where*

- (i) *C has finite elasticity.*
- (ii) *$T_2(C)$ has infinite elasticity.*

Proof. Let $\Sigma = \{a, b\}$ and let C_C be the collection of all languages L_n , where, for all $n \geq 0$, $L_n = \{ba^{n+1}b, bb\}$. It is not hard to see that C_C has finite elasticity. To see that $T_2(C_C)$ has infinite elasticity, choose the sequence of strings $w'_0 = bab\#bb$, $w'_1 = baab\#bb$, \dots and the sequence of languages $L'_0 = T_2(L_1)$, $L'_1 = T_2(L_2)$, \dots \square

However, for a slight variation of the language class $T_2(C)$, namely the class $T_4(C) = \{L \circ \Sigma^* \mid L \in C\}$, it can be shown that $T_4(C)$ has finite elasticity in case C has finite elasticity.

Proposition 5 *Let C be an indexable class that has finite elasticity. Then, $T_4(C)$ has also finite elasticity.*

Hence, $T_4(C) \in \text{LimTtxt}$.

Now, the latter theorem can be invoked to show that $T_2(C) \in \text{LimTtxt}$ in case that C has finite elasticity.

Theorem 6 *Let C be an indexable class that has finite elasticity. Then, $T_2(C) \in \text{LimTtxt}$.*

Finally, there is another, somehow orthogonal, sufficient condition that ensures that one can solve learning problems of type 2.

Theorem 7 *Let C be an indexable class that only contains finite languages. Then, $T_2(C) \in \text{LimTtxt}$.*

References

- Angluin, D. 1980. Inductive inference of formal languages from positive data. *Information and Control* 45:117–135.
- Gold, M. E. 1967. Language identification in the limit. *Information and Control* 14:447–474.
- Grieser, G.; Jantke, K. P.; Lange, S.; and Thomas, B. 2000. A unifying approach to HTML wrapper representation and learning. In *Proc. Conference on Discovery Science*, LNAI 1967, 50–64. Springer–Verlag.
- Motoki, T.; Shinohara, T.; and Wright, K. 1991. The correct definition of finite elasticity: corrigendum to Identification of unions. In *Proc. Workshop on Computational Learning Theory*, 375. Morgan Kaufmann Publishers
- Thomas, B. 1999a. Anti-unification based learning of T-Wrappers for information extraction. In *Proc. AAAI Workshop on Machine Learning for IE*, 15–20.
- Thomas, B. 1999b. Logic programs for intelligent web search. In *Proc. Symposium on Methodologies for Intelligent Systems*, LNAI 1609, 190–198. Springer–Verlag.
- Wright, K. 1989. Identification of unions of languages drawn from an identifiable class. In *Proc. Workshop on Computational Learning Theory*, 328–333. Morgan Kaufmann Publishers
- Zeugmann, T., and Lange, S. 1995. A guided tour across the boundaries of learning recursive languages. In *Algorithmic Learning for Knowledge-Based Systems*, LNAI 961, 190–258. Springer–Verlag.