Formal Learning Theory

Menno van Zaanen   Colin de la Higuera

1 Introduction

When dealing with language, (machine) learning can take many different faces, of which the most important are those concerned with learning languages and grammars from data. Questions in this context have been at the intersection of the fields of inductive inference and computational linguistics for the past fifty years. To go back to the pioneering work, Chomsky (1955) and Solomonoff (1964) were interested, for very different reasons, in systems or programs that could deduce a language when presented information about it.

Gold (1967) proposed a little later a unifying paradigm called identification in the limit, and the term of grammatical inference seems to have appeared in Horning's (1969) PhD thesis.

Out of the field of linguistics, researchers and engineers dealing with pattern recognition, under the impulsion of Fu (1974), invented algorithms and studied subclasses of languages and grammars from the point of view of what could or could not be learned (Fu and Booth, 1975).

Researchers in machine learning tackled related problems (the most famous being that of inferring a deterministic finite automaton, given examples and counter-examples of strings). Angluin (1981, 1987) introduced the important setting of active learning, or learning from queries, whereas Pitt and Warmuth (1993) and Pitt (1989) gave several complexity inspired results, exposing the hardness of the different learning problems.

In more applied areas, such as computational biology, researchers also worked on learning grammars or automata from strings, e.g. Brazma et al. (1998). Similarly, stemming from computational linguistics, one can point out the work relating language learning with more complex grammatical formalisms (Kanazawa, 1998), the more statistical approaches based on building language models, or the different systems introduced to automatically build grammars from sentences (van Zaanen, 2000; Adriaans, 1992). Surveys of related work in specific fields can be found in Sakakibara (1997); de la Higuera (2005); Wolf (2006).

When considering the history of formal learning theory, several trends can be identified. From "intuitive" approaches described in early research, more fundamental ideas arose. Based on these ideas and a wider availability of data, more research was directed into applied language learning. Recently, there has been a trend asking for more theoretically founded proofs in the applied area, mainly due to the increasing size of the problems and the importance of having guarantees over the results. These trends have led to the highly interdisciplinary character of formal language learning. Aspects of natural language learning (as an application arena), machine learning, and information theory can all be found here.

When attempting to find the common features of work in the field of language learning, one should at least consider two dimensions. Learning takes place in a *setting*. Issues in this dimension are properties of training data, such as positive/negative instances, amount, or noise levels, but also the measure of success. The other dimension deals with *paradigms* with respect to generalization over the training data. The goal of language learning is to find the language that is used to generate the training data. This language is typically more general than the training data, requiring a generalization approach.

This chapter is organized along the learning setting and paradigms dimensions. Firstly, we will look at different learning settings and their parameters. Secondly, different learning paradigms are discussed, followed by a conclusion.

2 Settings

The task of language learning deals with finding a language given a sample taken from that language. Typically, this language is described using a grammar, which is a compact, finite representation of a possibly infinite language. This general description of language learning leaves many questions open with two main questions in particular. What does the sample look like? When are we successful in learning the language? In the next sections we will concentrate on possible answers to these questions.

2.1 Learning settings

To describe a learning setting we can discuss the sort of information we have access to, the process that is allowing us to access this information and the quality of this information. These are parameters that define the learning settings.

In general, language learning is performed given strings sampled from a language. Alternatively, both examples and counter-examples, i.e. strings with the appropriate labels can be provided. This is called the *positive versus positive and negative input*, and corresponds to *learning from an informant.*

The examples the learner receives from the language may be in the shape of plain sequences, but the strings may also arrive with additional information, such as brackets or tags. This provides the learner with *structured input* (*learning from text*).

Additional information may also come from an underlying distribution over the strings that might be stable enough over time. This kind of additional information is called *distributional input.*

The learner may have interaction with the environment and ask for the status of some particular string or for some external expertise, which allows for *exploration of the input.*

Finally, the strings may have be corrupted, indicating the presence of *noise*.

2.1.1 Positive versus positive and negative input

Given data, a language learning system is supposed to make some hypothesis as to what the corresponding language may be. From the view of positive and negative input, there are two ways the system may receive information about the language. In the first setting, which is usually called *learning from text*, the learner is given only examples of strings in the language. In the second setting, called *learning from an informant*, the learner is also given some negative examples, or counter-examples.

The first situation is considered closer to natural language acquisition problems. The child is being talked to and only positive instances of the language are used[1]. The second situation is closer to a pattern recognition or classification task, where we must learn to separate one class (the language) from the complement.

2.1.2 Structured input

In some cases, the data will not just be raw strings, but will also contain some form of structure. Various levels of help are possible. In the simplest form, sequences contain some partial bracketing. More information is contained when the bracketing is complete. In this case, the string can be read as a tree whose leaves are the symbols in the string and whose internal nodes are unlabeled. More information is provided when the nodes of the tree are labelled. Clearly, when the data contains more annotation, the learner will have more information to build its hypothesis from.

2.1.3 Distributional input

If we suppose that the data has been obtained through sampling, then there is an underlying probability distribution over the strings. In most cases, however, we do not have a description of this distribution. We identify three approaches that take this into account.

The first approach assumes that the data is sampled according to an unknown distribution, and that what we learn will be measured with respect to the unknown distribution. This corresponds to the well-known PAC-learning (Probably Approximately Correct) setting (Valiant, 1984).

The second approach assumes that the data is sampled according to a distribution itself defined by a grammar or an automaton. The goal is now no longer to classify strings, but to learn this distribution. The learning process can be evaluated either by accepting a small error (which happens most of the time, since a particular sampling could have been corrupted), or in the limit, with probability one (Carrasco and Oncina, 1994).

[1]Note that the amount of information provided to children is also under discussion (Sokolov and Snow, 1994).

One can even hope for a combination of both these criteria, which is the third approach. A *probabilistic language*[2] $\mathcal{D}$ is a probability distribution over $\Sigma^\star$. The probability of a string $x \in \Sigma^\star$ under the distribution $\mathcal{D}$ is denoted by a positive value $Pr_{\mathcal{D}}(x)$, where $\sum_{x\in\Sigma^\star} Pr_{\mathcal{D}}(x) = 1$. If the distribution is modelled by some syntactic machine $\mathcal{A}$, the corresponding probability distribution is denoted $Pr_{\mathcal{A}}$ (de la Higuera and Oncina, 2004).

2.1.4 Exploration of the input

It is argued by a number of authors that (child or natural) language acquisition is not a one directional process. For instance, a child can *interact* with its mother through a number of means. Having access to an expert may help learning.

A framework to study learning through interaction with the teacher, or oracle, is that of active or query learning (Angluin, 1987). In this setting, a learner can query an oracle by means of questions in a predefined format. Typically, the learner may make a membership query (a string is submitted and the oracle labels it), or an equivalence query (a machine is submitted and the oracle answers whether the corresponding language is the target or not). Many alternative queries have been defined (Angluin, 2004). Recently, correction queries have been introduced (Becerra-Bonache and Yokomori, 2004; Becerra-Bonache et al., 2008). If the query string is not in the target language, a correction inside the language is provided.

2.1.5 Noise

There are many situations where the data is noisy. For example, in a speech recognition task, there are many situations where transcription can have gone wrong, where the sentences do not follow the ideal grammar one is hoping to learn, or where a variety of utterances are possible. Transcription problems can lead to data sets that, without being corrupt, are far away from the idealized situation one encounters in formal language theory (Tantini et al., 2006).

2.2 Evaluation settings

When learning a language given a sample, a selection has to be made from many languages that fit the sample. In order to evaluate effectiveness of learning, a measure of success has to be defined. In this section different measures of success in learning will be considered.

A distinction can be (and often is) made between formal language learning in Sections 2.2.1–2.2.3 (aiming to proof learnability of classes of languages and therefore yielding clarity and formalism) and empirical learning in Section 2.2.5 (aiming to build working models of naturally occurring data, addressing practical learning situations). These two fields have been and still are relatively dis-

[2]We denote by $\boldsymbol{\mathcal{L}}$ a class of languages. $\boldsymbol{\mathcal{G}}$ is a class of representation of objects in this class and $\mathbb{L} : \boldsymbol{\mathcal{G}} \to \boldsymbol{\mathcal{L}}$ is the *naming function*, *i.e.* $\mathbb{L}(G)$ (with $G \in \boldsymbol{\mathcal{G}}$) is the language denoted, accepted, recognized or represented by G. As usual, Σ is the alphabet of the language.

tinct. It seems more work is necessary to define models that are both practically applicable and at the same time provide users with mathematical guarantees. Research in this direction is mentioned in Section 2.2.4.

2.2.1 Identification in the limit

Identification in the limit (Gold, 1967) describes a process where learning is a never ending process. The learner is given information, builds a hypothesis, receives more information, updates the hypothesis, and so on. This setting may seem an unnatural process, completely abstract, and incapable of corresponding to concrete learning situation. However, it provides some useful insights.

Firstly, identification in the limit requires the notion of target. A target language, from which the data is extracted, pre-exists to the learning process.

Secondly, even if more typical learning situations are not incremental, it is still worth studying a learning session as part of a process. It may be that at some particular moment the learning process returned a good hypothesis, but unless we have some completeness guarantee about the data we have got, there is no way we can be sure. In other words, one can study the fact that we are learning a language but not that we have finished learning one.

Thirdly, even knowing that our algorithm does identify in the limit, does not give us any guarantee on a specific situation. What we do know is that if the algorithm does not identify in the limit, there is necessarily a *hidden bias*. There is at least one possible language which, for some unknown or undeclared reason, is not learnable.

The advantage of using an algorithm that has the propriety of identifying in the limit is that you can argue, if the algorithm fails: *"Don't blame me, blame the data"* (de la Higuera, 2006).

Definition 1 *Let* $\boldsymbol{\mathcal{L}}$ *be a class of languages. A* presentation *is a function* $\boldsymbol{f} : \mathbb{N} \to \boldsymbol{X}$ *where* $\boldsymbol{X}$ *is some set. The set of all admitted presentations for* $\boldsymbol{\mathcal{L}}$ *is denoted by* $\boldsymbol{Pres}(\boldsymbol{\mathcal{L}})$ *which is a subset of* $\left[\mathbb{N} \to \boldsymbol{X}\right]$.

In some way these presentations denote *languages from* $\boldsymbol{\mathcal{L}}$*, i.e. there exists a function* $\textsc{Yields} : \boldsymbol{Pres}(\boldsymbol{\mathcal{L}}) \to \boldsymbol{\mathcal{L}}$. *If* $L = \textsc{Yields}(\boldsymbol{f})$ *then we will say that* $\boldsymbol{f}$ *is a presentation of* L. *We also denote by* $\boldsymbol{Pres}(L)$ *the set* $\{\boldsymbol{f} \in \boldsymbol{Pres}(\boldsymbol{\mathcal{L}}) : \textsc{Yields}(\boldsymbol{f}) = L\}$.

A presentation of a language has to be meaningful. In other words, we should be able to associate presentations with a unique language which means that for the setting to be valid we require that $\textsc{Yields}(\boldsymbol{f}) = \textsc{Yields}(\boldsymbol{g})$.

We summarize these notions in Figure 1. The general goal of learning (in the framework of identification in the limit) is to find a learning algorithm $\mathbf{A}$ such that $\forall \boldsymbol{f} \in \mathbf{Pres}(\boldsymbol{\mathcal{L}}), \exists n \in \mathbb{N} : \ \forall m \geq \ n, \mathbb{L}(\mathbf{A}(\boldsymbol{f}_m)) = \textsc{Yields}(\boldsymbol{f})$. When this is true, we will say that class $\boldsymbol{\mathcal{L}}$ is identifiable in the limit by presentations in $\mathbf{Pres}(\boldsymbol{\mathcal{L}})$.

Gold (1978) proved the first and essential results in this setting:

$$\begin{tikzcd}
\mathbf{Pres}(\boldsymbol{\mathcal{L}}) \arrow[rrrr, phantom, "\textsc{Yields}"] & & & & \boldsymbol{\mathcal{L}} \\
& \mathbf{A} & & \mathbb{L} & \\
& & \boldsymbol{\mathcal{G}} & &
\end{tikzcd}$$

Figure 1: The learning setting.

Theorem 1 *Any recursively enumerable class of recursive languages is identifiable in the limit from an informant and no super-finite class of languages is identifiable in the limit from text.*

A *super-finite* language class is a class that contains all finite languages and at least one infinite language. Of course, the theorem holds for the usual classes of languages from the Chomsky hierarchy.

2.2.2 Identification in the limit with probability one

Let us suppose that we have a distribution over the set $\Sigma^\star$ of all strings. If we can identify this distribution, we can predict the next string, or the next symbol in a string. For this, we need to extend the results of identification in the limit to the probabilistic case. In this case, we assume that the distributions are generated by probabilistic machines (Vidal et al., 2005).

In the framework of identification in the limit with probability one, the learner is given an increasing sequence of strings and identification can only be avoided for a finite period of time. Just like in the case of identification in the limit, the actual moment at which identification will be achieved is usually not guaranteed.

The analysis in this setting of the well-known algorithm Alergia is given in Carrasco and Oncina (1999). A variety of alternative settings are scrutinized by de la Higuera and Oncina (2004).

2.2.3 Pac-learning

In a probabilistic setting, it is unclear whether we need exact identification. The distribution gives us a notion of distance to the target, which can be used to define "not too far" identification. Since we do not have full control over the sampling process we can only hope that in most cases (probably) we will be approximately correct. This is the Pac (Probably Approximately Correct) framework (Valiant, 1984).

Pac-learning grammars We suppose there is an unknown distribution under which strings can be sampled. After learning, we hope to end up with an ϵ-good hypothesis:

Definition 2 (ϵ-good hypothesis) *Let G be the target grammar and H be a hypothesis grammar over Σ. Let $\mathcal{D}$ be a distribution over $\Sigma^\star$. For $\epsilon > 0$, H is an* ϵ-good hypothesis *with respect to G if $Pr_{\mathcal{D}}\big(x \in \mathbb{L}(G) \oplus \mathbb{L}(H)\big) \leq \epsilon$.*

A learning algorithm is now asked to learn a grammar given a *confidence* parameter δ and an *error* parameter ϵ. The algorithm must also know an upper bound on the size of the target grammar and on the length of the samples.

The model has been adapted by taking into account only certain types of distributions and simple PAC-learning has been considered with more success (Denis et al., 1996).

Pac-learning distributions When trying to learn a probabilistic automaton or grammar (instead of a classifier) we require an algorithm that makes an error of less than ϵ with high probability after seeing just a polynomial number of examples. In this case, the problem is in keeping track of ϵ. Typically, one will define a distance over distributions and require that the distance between the target and the hypothesis be less than ϵ. Only recently, we have received some insight in the importance of selecting a specific distance. Exploratory publications on this topic are Thollard and Clark (2004).

2.2.4 Restrictions

The general setting described in the sections above is close to what has typically been called inductive inference. But if we want to apply the theoretical algorithms in practice, it is necessary to come up with algorithms whose resources are bounded in some way.

Bounding quantity of data Even if an *a priori* bound on the number of examples needed for a particular learning algorithm to converge is not going to mean anything (remember that we do not have any control over the sequence of examples) (Pitt, 1989), it is possible to have an optimistic point of view.

We can bound the size of a set, called the characteristic set, whose presence in the learning sample ensures identification. It is known that this quantity is polynomial for DFA but not for NFA nor for context-free grammars (de la Higuera, 1997).

Bounding time If we try to bound the overall time, the learning algorithm is only allowed to access a polynomial number of items of data before returning a correct hypothesis, or to say that the learner has update polynomial time: constructing $H_n = \mathbf{A}(\boldsymbol{f}_n)$ requires $\mathcal{O}(p(\|\boldsymbol{f}_n\|))$, where $p()$ is a polynomial.

However, neither definition is entirely convincing. In the first case, since we have no control over the actual presentation, we cannot avoid that the presentation starts with very long and uninteresting examples. In the second case, Pitt (1989) proves that identification can be postponed in such a way as to make any identification in the limit algorithm have polynomial update time.

Bounding memory Practically speaking it seems to be unreasonable to consider that one should learn and keep track of all the data. In fact, learning is about generalization, and hence allows forgetting. The question of studying learners that have access only to their previous hypothesis and to only some of the examples seen up to then has not received sufficient attention. These learners are called *memory-limited* scientists (Osherson et al., 1997).

Bounding prediction errors It is possible to put a bound on the number of times the learning algorithm may output wrong hypotheses before converging. The algorithm is therefore required to make a polynomial number of implicit prediction errors (Ipe), where an implicit prediction error is made whenever $\mathbf{A}(\boldsymbol{f}_{n-1}) \nvDash \boldsymbol{f}(n)$. An algorithm that changes its mind when a newly presented string is in error with respect to the current hypothesis is said to be *consistent*.

Identification in Ipe-*polynomial time* of $\boldsymbol{\mathcal{G}}$ takes place when there is a learning algorithm which 1. identifies $\boldsymbol{\mathcal{G}}$ in the limit, 2. has polynomial update time, and 3. makes a polynomial number of implicit prediction errors. Pitt (1989) showed that Dfa could not be identified in Ipe-*polynomial time*.

Bounding mind changes An alternative to counting the number of errors is that of counting the number of changes of hypothesis. In general, there is no reason why change is required, but combined with identification in the limit the definition makes sense. A mind change (Mc) is made whenever $\mathbf{A}(\boldsymbol{f}_n) \neq \mathbf{A}(\boldsymbol{f}_{n-1})$.

An algorithm that never changes its mind when the current hypothesis is consistent with the new presented element is said to be *conservative*.

Algorithm $\mathbf{A}$ is said to make a *polynomial number of mind changes*(Mc) if for any presentation of $L = \mathbb{L}(G)$ the number of mind changes is polynomial in $\|G\|$, and an algorithm $\mathbf{A}$ identifies a class $\boldsymbol{\mathcal{G}}$ in the limit in Mc-polynomial time if 1. $\mathbf{A}$ identifies $\boldsymbol{\mathcal{G}}$ in the limit, 2. has polynomial update time, and 3. $\mathbf{A}$ makes a polynomial number of mind changes.

2.2.5 Real world evaluation

The evaluation methods described so far assume that the underlying grammar to be learned is known. Learning grammars in a real world setting, where the underlying grammar is typically unknown, requires other evaluation techniques.

The evaluation methods can roughly be divided into four groups; van Zaanen (2002, pp. 58–62) describes three of them and van Zaanen et al. (2004) also mentions the fourth group. Here, we will discuss all groups of evaluation methods in more detail.

Looks-good-to-me In the *looks-good-to-me* approach, the output of the GI system is analyzed manually. Since the output of the system can be a fair amount, the analysis is often focussing on particular aspects of the GI system, such as whether recursive structures occur.

This approach as two main advantages. Firstly, only unstructured data is required as input, since the structured output is analyzed by hand. This makes it easy to apply the system to different data sets, for example, natural language for which no manually annotated corpora exist. Secondly, the evaluation can focus on specific structures. Not only can the output of the GI system be easily searched for this structure, the input of the system can be tailored to learning these structures as well.

The main disadvantage of this approach is that it can only provide a useful means of comparison of systems if the evaluation is performed by an independent expert (or experts) comparing outputs of rival systems at the same time.

In practice, however, several GI developers have applied the *looks-good-to-me* evaluation to their own systems, rather than perform objectively quantifiable comparisons. This gives rise to the name of the approach. Most often, according to this evaluation method the system seems to perform well.

It has to be noted that human evaluation of output is accepted standard practice in some other fields, for example, in machine translation evaluation where a range of translations may be equally valid (Elliott et al., 2003).

Rebuilding known grammars In this evaluation approach, one or more "toy" grammars are selected beforehand. These grammars are most often relatively small and typically have known properties, such as context-freeness. In practice, there are some grammars that are considered "standard" test grammars (Cook et al., 1976; Hopcroft et al., 2001).

Using these grammars sentences are generated, which are then used as input for the GI system. The output of the system (i.e. the grammar or the structured version of the input) is then compared against the original data or grammar.

There are two ways of testing for grammar equivalence: *language equivalence*, where the languages of two grammars should be the same, and *structural equivalence*, where the language as well as the analyzes, modulo the naming of non-terminal type labels are the same.

Whereas measuring structural equivalence is possible (automatically) by analyzing the grammar rules, evaluating language equivalence is undecidable. Whereas for finite languages all sentences can be generated and compared, this is not possible for infinite languages. This also explains scalability problems, resulting in the use of small artificial grammars.

There are other problems with this approach as well. Firstly, grammars can be selected specifically for the algorithm under consideration, allowing for unfair comparisons. Secondly, the way strings are generated from the grammar may have an influence on the learning as well. Instead of trying to learn the grammar, this may result in learning the distribution that is used to generate the strings. From the generation point of view, requirements need to be made as well, for instance, at some point all grammar rules have to be used or it may be necessary to restrict the generation method to limit the sentence length.

Compare against treebank The next approach evaluates GI systems based on information contained in an annotated treebank. This treebank is considered a "gold standard" and the full content is taken as correct. To evaluate a system, all structure is stripped from the treebank. The plain strings are given to the

GI system and the resulting structures are compared against the gold standard, which measures how well the GI system can find the original structure.

The gold standard may contain manually annotated data or structures generated by a grammar, allowing flexibility in the data or grammars used. Different natural languages or data from specific domains can be tested and like "standard" evaluation grammars, "standard" evaluation treebanks can be designed.

Note that GI systems may need to be adapted to generate structured versions of the input sentences. When GI systems output grammars, the test sentences need to be be parsed, which may introduce problems when the grammar is ambiguous. Compare this to the *rebuilding known grammars* evaluation approach, where the output of the GI system needs to be a grammar.

The main problem with this approach is that a collection of structured sentences is needed, which may not be a problem when evaluating known grammars, but when evaluating on natural languages, a treebank will need to be build manually (or semi-automatically).

Language membership This method has been used in GI competitions such as Abbadingo (Lang et al., 1998a), Gowachin, Omphalos (Starkie et al., 2005), and Tenjinno (Starkie et al., 2006)[3]. GI systems are tested according to their effectiveness to decide on language membership of test sentences[4].

The precision-oriented language membership evaluation method measures the effectiveness of the system to decide language membership. After training, the GI system should decide given a set of sentences which of these are in the language. The classifications can be correct (which means that the sentence was classified correctly) or incorrect. Precision is defined as the ratio of correctly classified sentences.

Measuring coverage can be done by generating sentences using the learned grammar. This shows how much of the original language is covered by the grammar and how many sentences outside the language can be generated. It is essential (just as with the computation of precision) that the sentences generated are distinct and different from the sentences in the training data.

Increased precision means that more sentences of the learned language are contained in the original language. With perfect precision, all sentences of the learned language are in the original language. Conversely, increasing recall means more sentences in the original language described by the learned language. Perfect recall occurs when all sentences of the target language are in the learned language.

Another metric, which can be used to describe the descriptive power of grammars, or language models in general, is perplexity. The perplexity of a string $a_1 \dots a_n$ is computed as 2^H with H being entropy $H = -\sum_{i=1}^{n} \frac{P(a_i) \log_2 P(a_i)}{n}$. This measures how well the probability distribution P, defined by the language model, predicts the elements in the string. Lower perplexity means that fewer

[3]In fact, these competitions only considered the precision metric in this context.

[4]Implemented as such, no actual learning of a grammar is required. A generic machine learning classifier may suffice.

bits are needed to describe the sequence.

3 Paradigms

Over the years different language learning approaches have been investigated. These approaches can be roughly divided into three different paradigms. The first two paradigms (state-merging and substitutability) start with a simple structure describing the samples and then use different approaches to generalize the structure. The third paradigm (counting) fixes the structure first and then learns weights on parts of the structure.

3.1 State-merging

State-merging algorithms start by building tree-like automata based on the training data. Next, pairs of states are chosen and a compatibility test is performed. Two states are compatible when merging them into a unique state results in a better automaton, i.e. consistent with the data and/or with some extra combinatorial properties. Since state merging can lead to non-determinism, more node merges, or a folding operation, might be needed.

The crucial issue how to choose the pairs to be tested, and, when various merges are possible, how to find the best one. Evidence driven methods (Lang et al., 1998b) use the quantity of resulting merges for this.

State-merging algorithms have been used to learn finite state automata and extensions of these algorithms for probabilistic automata (Carrasco and Oncina, 1994), Büchi automata (de la Higuera and Janodet, 2004), or transducers (Castellanos et al., 1993) have been proposed.

3.2 Substitutability

The idea behind substitutability-based approaches is that when substrings occur in the same context, they have properties in common. Such substrings are therefore combined in a cluster, which is denoted by a non-terminal.

Essentially, the idea is substitutability is derived from Harris (1951), who states: "If [elements] are *freely substitutable* for each other they are descriptively equivalent, in that everything we henceforth say about one of them will be equally applicable to the others." Harris also explains how substitutable elements can be found, which comes down to: unequal parts are substitutable.

More formally, consider two strings $w = lur$ and $x = lvr$. Applying substitutability results in $w = l(u)_X r$ and $x = l(v)_X r$. The evidence in the form of the strings leads to the assumption that both u and v can be generated from an underlying non-terminal X, which is used in the context of l and r. It may be clear that the idea of substitutability, as shown in these strings can be generalized in with multiple substitution in one pair of strings.

Alignment-Based Learning (ABL) (van Zaanen, 2000, 2002) makes direct use of the idea of substitutability. Comparing natural language sentences pairwise, structure is learned. A more efficient implementation based on suffix-trees is proposed in Geertzen and van Zaanen (2004). A similar system, EMILE, finds more structure by replacing all occurrences of substrings that have been grouped together with the corresponding non-terminal (Adriaans, 1992). More recently, the idea of substitutability has also led to formal proofs of learnability (Clark and Eyraud, 2007; Yoshinaka, 2008).

3.3 Counting

In contrast to the approaches discussed so far, we are often not necessarily concerned with the exact underlying structure. Instead, we would only like to know whether strings are elements of the language or not. To decide this, the shape of the structure is irrelevant.

There are several approaches that, instead of learning the structure, fix a structure beforehand and estimate probabilities on the structure. Well-known examples are the Baum-Welch algorithm, which finds unknown parameters of Hidden-Markov Models (Baum et al., 1970). This algorithm is a generalized version of the Expectation Maximization algorithm (Dempster et al., 1977). A similar algorithm, which can be used to estimate probabilities in a probabilistic context-free grammar is the Inside Outside algorithm (Lari and Young, 1991). More recently, another approach has been investigated, taking many possible (sub)structures concurrently into account and allowing statistics to select preferred structures (Bod, 2006).

4 Conclusion

The research performed in the area of formal learning theory deals with learning compact representations of languages. This can be tackled from many different points of view. Depending on specific settings, one may wonder whether a particular class of languages can be learned efficiently. We recognize different means of measure success which all lead to different learnability results.

In addition to the different settings and parameters, there is the dimension of paradigms. Many different approaches to language learning have been proposed.

In this chapter we have provided an overview of the most well-known settings and paradigms. This should provide enough information for people to investigate in more detail the existing literature on language learning in this field and related fields, such as computational linguistics, computational biology, or pattern recognition.

References

Adriaans, P. (1992). *Language Learning from a Categorial Perspective.* PhD thesis, University of Amsterdam, Amsterdam, the Netherlands.

Angluin, D. (1981). A note on the number of queries needed to identify regular languages. *Information and Control*, 51:76–87.

Angluin, D. (1987). Queries and concept learning. *Machine Learning Journal*, 2:319–342.

Angluin, D. (2004). Queries revisited. *Theoretical Computer Science*, 313(2):175–194.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171.

Becerra-Bonache, L., de la Higuera, C., Janodet, J. C., and Tantini, F. (2008). Learning balls of strings from edit corrections. *Journal of Machine Learning Research*, 9:1841–1870.

Becerra-Bonache, L. and Yokomori, T. (2004). Learning mild context-sensitiveness: Toward understanding children's language learning. In Paliouras and Sakakibara (2004), pages 53–64.

Bod, R. (2006). Unsupervised parsing with u-dop. In *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 85–92, Morristown, NJ, USA. Association for Computational Linguistics.

Brazma, A., Jonassen, I., Vilo, J., and Ukkonen, E. (1998). Pattern discovery in biosequences. In Honavar and Slutski (1998), pages 257–270.

Carrasco, R. C. and Oncina, J. (1994). Learning stochastic regular grammars by means of a state merging method. In Carrasco, R. C. and Oncina, J., editors, *Grammatical Inference and Applications, Proceedings of* ICGI *'94*, number 862 in LNAI, pages 139–150. Springer-Verlag.

Carrasco, R. C. and Oncina, J. (1999). Learning deterministic regular grammars from stochastic samples in polynomial time. RAIRO *(Theoretical Informatics and Applications)*, 33(1):1–20.

Castellanos, A., Vidal, E., and Oncina, J. (1993). Language understanding and subsequential transducer learning. In *International Colloquium on Grammatical Inference*, pages 11/1–11/10.

Chomsky, N. (1955). *The logical structure of linguistic theory.* PhD thesis, Massachusetts Institute of Technology.

Clark, A. and Eyraud, R. (2007). Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745.

Cook, C. M., Rosenfeld, A., and Aronson, A. R. (1976). Grammatical inference by hill climbing. *Informational Sciences*, 10:59–80.

de la Higuera, C. (1997). Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, 27:125–138.

de la Higuera, C. (2005). A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332–1348.

de la Higuera, C. (2006). Data complexity issues in grammatical inference. In Basu, M. and Ho, T. K., editors, *Data Complexity in Pattern Recognition*, pages 153–172. Springer-Verlag.

de la Higuera, C. and Janodet, J.-C. (2004). Inference of ω-languages from prefixes. *Theoretical Computer Science*, 313(2):295–312.

de la Higuera, C. and Oncina, J. (2004). Learning probabilistic finite automata. In Paliouras and Sakakibara (2004), pages 175–186.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Denis, F., d'Halluin, C., and Gilleron, R. (1996). PAC learning with simple examples. In *13th Symposium on Theoretical Aspects of Computer Science, Stacs '96*, Lncs, pages 231–242.

Elliott, D., Hartley, A., and Atwell, E. (2003). Rationale for a multilingual aligned corpus for machine translation evaluation. In Archer, D., Rayson, P., Wilson, A., and McEnery, T., editors, *Proceedings of the Corpus Linguistics 2003 conference; Lancaster, UK*, pages 191–200.

Fu, K. S. (1974). *Syntactic Methods in Pattern Recognition*. Academic Press, New-York.

Fu, K. S. and Booth, T. L. (1975). Grammatical inference: Introduction and survey. Part I and II. Ieee *Transactions on Syst. Man. and Cybern.*, 5:59–72 and 409–423.

Geertzen, J. and van Zaanen, M. (2004). Grammatical inference using suffix trees. In Paliouras and Sakakibara (2004), pages 163–174.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10(5):447–474.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, 37:302–320.

Harris, Z. S. (1951). *Structural Linguistics*. University of Chicago Press, Chicago:IL, USA and London, UK, 7th (1966) edition. Formerly Entitled: Methods in Structural Linguistics.

Honavar, V. and Slutski, G., editors (1998). *Grammatical Inference, Proceedings of* Iccgi *'98*, number 1433 in Lnai. Springer-Verlag.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2001). *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Company, Reading:MA, USA.

Horning, J. J. (1969). *A study of Grammatical Inference*. PhD thesis, Stanford University.

Kanazawa, M. (1998). *Learnable Classes of Categorial Grammars*. CSLI Publications, Stanford, Ca.

Lang, K., Pearlmutter, B. A., and Coste, F. (1998a). The Gowachin automata learning competition.

Lang, K. J., Pearlmutter, B. A., and Price, R. A. (1998b). Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In Honavar and Slutski (1998), pages 1–12.

Lari, K. and Young, S. J. (1991). Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 5:237–257.

Osherson, D., de Jongh, D., Martin, E., and Weinstein, S. (1997). *Handbook of Logic and Language*, chapter Formal learning theory, pages 737–775. MIT Press.

Paliouras, G. and Sakakibara, Y., editors (2004). *Grammatical Inference: Algorithms and Applications, Proceedings of* ICGI *'04*, volume 3264 of LNAI. Springer-Verlag.

Pitt, L. (1989). Inductive inference, DFA's, and computational complexity. In *Analogical and Inductive Inference*, number 397 in LNAI, pages 18–44. Springer-Verlag.

Pitt, L. and Warmuth, M. (1993). The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the Association for Computing Machinery*, 40(1):95–142.

Sakakibara, Y. (1997). Recent advances of grammatical inference. *Theoretical Computer Science*, 185:15–45.

Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., and Tomita, E., editors (2006). *Grammatical Inference: Algorithms and Applications, Proceedings of* ICGI *'06*, volume 4201 of LNAI. Springer-Verlag.

Sokolov, J. K. and Snow, C. E. (1994). The changing role of negative evidence in theories of language development. In Gallaway, C. and Richards, B. J., editors, *Input and Interaction in Language Acquisition*, pages 38–55. Cambridge University Press, Cambridge, UK.

Solomonoff, R. (1964). A formal theory of inductive inference. *Information and Control*, 7(1):1–22 and 224–254.

Starkie, B., Coste, F., and van Zaanen, M. (2005). Progressing the state-of-the-art in grammatical inference by competition. *AI Communications*, 18(2):93–115.

Starkie, B., van Zaanen, M., and Estival, D. (2006). The tenjinno machine translation competition. In Sakakibara et al. (2006), pages 214–226.

Tantini, F., de la Higuera, C., and Janodet, J.-C. (2006). Identification in the limit of systematic-noisy languages. In Sakakibara et al. (2006), pages 19–31.

Thollard, F. and Clark, A. (2004). PAC-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142.

van Zaanen, M. (2000). ABL: Alignment-Based Learning. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING); Saarbrücken, Germany*, pages 961–967. Association for Computational Linguistics.

van Zaanen, M. (2002). *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD thesis, University of Leeds, Leeds, UK.

van Zaanen, M., Roberts, A., and Atwell, E. (2004). A multilingual parallel parsed corpus as gold standard for grammatical inference evaluation. In Kranias, L., Calzolari, N., Thurmair, G., Wilks, Y., Hovy, E., Magnusdottir, G., Samiotou, A., and Choukri, K., editors, *Proceedings of the Workshop: The Amazing Utility of Parallel and Comparable Corpora; Lisbon, Portugal*, pages 58–61.

Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., and Carrasco, R. C. (2005). Probabilistic finite state automata – part I and II. Pattern Analysis and Machine Intelligence.

Wolf, G. (2006). *Unifying computing and cognition*. Cognition research.

Yoshinaka, R. (2008). Identification in the limit of k, l-substitutable context-free languages. In Clark, A., Coste, F., and Miclet, L., editors, *Grammatical Inference: Algorithms and Applications, Proceedings of* ICGI *'08*, volume 5278 of LNCS, pages 266–279. Springer-Verlag.