Artif Intell Rev (2011) 36:1–27
DOI 10.1007/s10462-010-9199-1

A survey of grammatical inference methods for natural language learning

Arianna D'Ulizia · Fernando Ferri · Patrizia Grifoni

Published online: 6 January 2011

Abstract The high complexity of natural language and the huge amount of human and temporal resources necessary for producing the grammars lead several researchers in the area of Natural Language Processing to investigate various solutions for automating grammar generation and updating processes. Many algorithms for Context-Free Grammar inference have been developed in the literature. This paper provides a survey of the methodologies for inferring context-free grammars from examples, developed by researchers in the last decade. After introducing some preliminary definitions and notations concerning learning and inductive inference, some of the most relevant existing grammatical inference methods for Natural Language are described and classified according to the kind of presentation (if text or informant) and the type of information (if supervised, unsupervised, or semi-supervised). Moreover, the state of the art of the strategies for evaluation and comparison of different grammar inference methods is presented. The goal of the paper is to provide a reader with introduction to major concepts and current approaches in Natural Language Learning research.

Keywords Grammatical inference · Natural language · Context free grammar

1 Introduction

Grammatical inference (also known as grammar induction, or grammar learning) deals with idealized learning procedures for acquiring grammars on the basis of exposure to evidence about languages (Pullum 2003).

Given the complexity of natural language (NL) and the high amount of human and temporal resources necessary for producing the grammars, the automation of grammar generation is and will continue to be one of the major areas of research in Natural Language Processing (NLP). To this aim, many grammar inference algorithms have been developed in the

A. D'Ulizia · F. Ferri · P. Grifoni (✉)
Consiglio Nazionale delle Ricerche – Istituto di Ricerche sulla Popolazione e le Politiche Sociali,
Via Palestro 32, 00185 Rome, Italy
e-mail: patrizia.grifoni@irpps.cnr.it

literature for natural language. These algorithms have been differently classified by several authors, according to the various features of the inference process. Roberts and Atwell (2002) provided a classification based on the underlying learning paradigm into: models based on Categorial Grammar, memory-based learning models, evolutionary computing models, and string-pattern searches. Edelman et al. (2005) classified grammar induction methods according to the type of input, splitting them in methods that learn from tagged corpora and methods that learn from raw (untagged) corpora. Analogously, Cramer (2007) classified the existing approaches in tag-based and word-based methods.

In this paper, a structured overview of the existing grammar inference methods for natural language is given. This survey provides a comprehensive summary of the present state of the literature, considering a wide scope of grammar inference methods. A classification of these methods is provided, based on the presentation set (e.g., text and informant) and the type of information (e.g., supervised, unsupervised and semi-supervised) they use for learning the grammar. A comparative review of the grammar inference methods, in light of the classification proposed, is given. Moreover, in order to identify the future trends of these methods, an analysis of their temporal evolution and the computational techniques applied to perform the language learning is provided.

Compared to the previous classification works, we review a much wider number of grammar inference algorithms and focus much more on the evaluation of these methods, performed by the authors using one of the three main evaluation strategies, that are the "*Looks-Good-to-me*", the "*Compare Against Treebank*" and the "*Rebuilding Known Grammars*". The goal of this review is to provide a reader who may not be very familiar with Natural Language Learning with introduction to major concepts and current approaches in this research.

The remainder of the paper is organized as follows. Section 2 is an introduction of some basic notions from language learning. Section 3 is dedicated to review the most relevant grammar inference methods for natural language developed in the last decade. Section 4 overviews the evaluation techniques usually applied for grammar inference, and describes how they have been applied to evaluate and compare the investigated grammar inference methods. Finally, Section 5 concludes the paper.

2 Basic notions from grammars and language learning

This section provides an introduction to the language learning issues, including the definition of context-free grammars, the most popular kind of grammar that has been firstly used to define the syntax of natural language, as well as a brief review of some preliminary notions from the language learning theory.

2.1 Context-free grammars

Context-free grammars (CFGs) were firstly defined by Chomsky in the mid-1950s (Chomsky 1957). A grammar is context-free when the expansion of a symbol does not depend on its context (i.e., the position of the symbol in a sequence or the relationship with surrounding symbols). A context-free grammar consists of four components:

T, is a finite set of terminal symbols;
N, is a finite set of non − terminal symbols;
P, is a finite set of production rules;
X, is a start symbol in N.

S → NP VP	A sentence (S) is a noun phrase (NP) plus a verb phrase (VB)
NP → N	A noun phrase is a noun
VP → V PP	A verb phrase is a verb plus a prepositional phrase (PP)
PP → PREP NP	A prepositional phrase (PP) is a preposition plus a noun phrase
N → *John* N → *CNR*	'John' and 'CNR' are nouns
PREP → *at*	'at' is a preposition
V → *works*	'works' is a verb

Fig. 1 Example of production rules of a CFG for a small fragment of English

Terminal symbols are the words that constitute the alphabet of the language (represented in italics in the subsequent examples).

Non-terminal symbols represent the grammatical categories, such as sentence (in short *S*), noun phrase (in short *NP*), verb phrase (in short *VP*), prepositional phrase (in short *PP*), determiner (in short *DET*), noun (in short *N*), verb (in short *V*), preposition (in short *PREP*), etc.

A production rule consists of a single non-terminal symbol, followed by an arrow → that is followed by a finite sequence of terminal and/or non-terminal symbols. Production rules express how different grammatical categories can be built up.

Any sequence of terminal symbols derived from the start symbol using production rules is called sentence. The set of sentences that can be derived from the start symbol applying the set of production rules constitutes the language generated by the grammar.

An example of production rules of a CFG for a small fragment of English is described in Fig. 1:

Let us now consider the string of words "*John works at CNR*". This is a correct sentence in the language defined by the grammar, since the sequence of terminal symbols "*John*" "*works*" "*at*" "*CNR*" can be derived from the start symbol *S* by repeatedly applying the production rules shown in Fig. 1.

Actually, natural language is characterized by a large variety of linguistic phenomena, which are not completely represented by CFGs, but need of the expressiveness of the class of context-sensitive grammars[1]. The name context-sensitive comes from the fact that the expansion of a symbol depends on its context (i.e., the position of the symbol in a sequence or the relationship with surrounding symbols).

Context-sensitive grammars, however, have two shortcomings with respect to natural language processing:

– parsing complexity: all known algorithms for parsing these grammars have exponential time complexity.
– too needless expressiveness: only a few linguistic phenomena, such as cross-serial dependency (that can occurs in Dutch and Swiss-German languages), require the expressiveness of context-sensitive grammars.

Consequently, although CFGs have less expressive power than context-sensitive ones, they are able to model all frequent linguistic phenomena of natural language assuring, at the same

[1] A context-sensitive grammar is a formal grammar $G = (\mathtt{T}, \mathtt{N}, \mathtt{P}, \mathtt{X})$ in which every production rule $p \in \mathtt{P}$ is of the form $wA\beta \rightarrow w\gamma\beta$, with $A \in \mathtt{T}$ (i.e., A is a non-terminal symbol) and w and $\beta \in (\mathtt{T} \cup \mathtt{N})^*$ (i.e., w and β are strings of non-terminals and terminals) and $\gamma \in (\mathtt{T} \cup \mathtt{N})^+$ (i.e., γ is a nonempty string of non-terminals and terminals).

time, a lower parsing complexity. For these reasons, the majority of grammars for natural language has been developed opting for context-free grammars instead of context-sensitive ones.

2.2 Language learning

All the main research studies in language learning agree with the fact that the learnability of various language classes, either in the Chomsky hierarchy (i.e. regular languages, context-free languages, context-sensitive languages, and unrestricted languages) or not, is a hard problem (i.e. there exists no polynomial time algorithm that can learn the target grammar from an arbitrary set of labeled examples).

The majority of the language learning models studied in the literature takes as input an initial set of positive training examples (i.e. sentences that should be recognized by the grammar) and produces in output the language description, i.e. the specific grammar able to recognize only these examples. To achieve that, a set of negative examples (i.e. sentences that should not be recognized by the grammar) is also needed for limiting the extent of generalisation, as an overly general grammar will never be refuted considering a new positive example.

Therefore, the two main issues that grammar inference methodologies have to face are the over-specialisation (or over-fitting) and the over-generalisation. The former occurs when the inference process produces a grammar whose language is smaller than the unknown target language (which is always the case when algorithms are not trained ad infinitum). This issue can be prevented by some extent setting aside some data (which takes part of the so-called "validation set") and measuring the performance on this data after each training example has been processed. The latter occurs when the inference process produces a grammar whose language is larger than the unknown target language. Over-generalisation can be controlled by using a set of negative examples.

One of the main results in language learning theory was reached by Gold (1967) in the middle 1960s. He proved that context-free grammars are not learnable from positive examples only. However, Gold's theorem does not cover all kinds of CFGs, such as for example stochastic CFGs and finite grammars that are indeed learnable, as showed by Horning (1969) and Adriaans (1992), respectively.

2.2.1 The presentation set

Following Gold's seminal paper (Gold 1967), the specification of a learning algorithm requires the definition of:

- the class of languages $\mathcal{L}$ to be inferred;
- the language description (or hypothesis) class $\mathcal{H}$ used to describe the languages in $\mathcal{L}$, which corresponds to the grammar in our case. Let $h \in \mathcal{H}$, $\mathcal{L}(h)$ denotes the language described by h;
- the way the learning process obtains information.

Gold models a learning algorithm $\mathcal{L}_{\mathcal{A}}$ as a function that takes as input a finite sequence of examples and gives as output a language description.

A *presentation* is an infinite sequence of examples. Two kinds of presentations are usually allowed:

- A *text* for a language $\mathcal{L}$ is an infinite sequence of strings $x_1, x_2, \ldots$from $\mathcal{L}$ such that every string of $\mathcal{L}$ occurs at least once in the text. The inference algorithms that use this type of

information are said to learn from *positive examples*. Note that the class of all the possible text presentations for a language $\mathcal{L}$ is denoted by $\mathcal{P}_L$.
– An *informant* for a language L is an infinite sequence of pairs $(x_1, d_1), (x_2, d_2), \ldots$ in $L \times \mathbb{B}$, (where $\mathbb{B}$ is the set of Booleans) such that every string of $\mathcal{L}$ occurs at least once in the sequence and $d_i = \text{true} \Leftrightarrow x_i \in \mathcal{L}$. The inference algorithms that use this type of information are said to learn from positive and negative examples. Note that the class of all the possible informant presentations for a language $\mathcal{L}$ is denoted by $\mathcal{PN}_{\mathrm{L}}$.

In NLP, large sets of positive examples may be available but it is rarely possible to obtain a set of negative examples for learning. To overcome this lack of negative evidence, the following two solutions have been proposed in the literature:

– to restrict the language to one of the classes of formal languages, which have been proven to be learnable from positive examples only, such as reversible languages (Angluin 1982), k-testable languages (Garcia and Vidal 1990), code regular and code linear languages (Emerald et al. 1996), pure context-free languages (Koshiba et al. 1997) and strictly deterministic automata (Yokomori 1995).
– to introduce various heuristics aiming to avoid over-generalisation without the use of negative examples, such as simplicity (Langley and Stromsten 2000).

In this survey, we take into account the kind of presentation (if text or informant) to broadly classify the grammar inference methods for natural language into *informant-based methods*, which learn from positive and negative examples, and *text-based methods*, which learn from positive examples only.

2.2.2 The type of information

In the model proposed by Gold (1967), the learning process involves two parties, the Learner and the Challenger (also called Teacher). The former is the party that has to identify the language, while the latter has to give to the learner examples of (unstructured) sentences taken from the language. In addition, a critic (also called an oracle) may be used by the learner for verifying if a certain sentence is a valid sentence in the language, i.e. it can be derived from the start symbol applying the set of production rules.

According to the Learner and the Challenger role and the use of a critic, language learning methods are classified in *supervised, unsupervised*, and *semi-supervised*.

Language learning methods are said to be *supervised* if they use a challenger to provide the examples of (unstructured) sentences and a critic to validate hypotheses about the language. This means that the method defines a connection between one set of sentences, called inputs and given by the challenger, and another set of sentences, called outputs and given by the critic. Generally, a learning method that uses a treebank or a structured corpus is a supervised method, as the structure of the sentences in the corpus can be considered the critic.

Language learning methods are said to be *unsupervised* if they only use a challenger. This means that they do not receive information from the critic about the structure of valid sentences in the language, and therefore they do not know what the output should look like.

A third class of language learning methods, called *semi-supervised* learning, is halfway between supervised and unsupervised learning. Semi-supervised learning methods use available unstructured data to improve supervised learning tasks when the structured data are scarce or expensive.

Therefore, in this survey, we take into account the type of information (if supervised, unsupervised, or semi-supervised) to broadly classify the grammatical inference methods

Table 1 Current grammar inference methods for NL and their classification

	Presentation set		Type of information		
	Text	Informant	Supervised	Unsupervised	Semi-supervised
ADIOS	X			X	
EMILE		X	X		
e-GRIDS	X			X	
CLL	X			X	
CDC	X			X	
INDUCTIVE CYK		X		X	
LAgtS	X			X	
GA-based		X	X		
ALLis	X		X		
ABL	X			X	
UnsuParse	X			X	
Incremental parsing	X			X	
Self-training	X				X
Co-training	X				X

for natural language into *supervised methods*, which use a treebank or a structured corpus, *unsupervised methods*, which do not have knowledge of the structure of the language, and *semi-supervised methods*, which combine unstructured linguistic data with small structured training sets.

3 Grammatical inference methods for natural language

The main research studies in grammatical inference have been made in various application domains, such as speech recognition (Baker 1979), computational linguistics (Adriaans 1992), computational biology (Sakakibara et al. 1994; Salvador and Benedı 2002), and machine learning (Sakakibara 1997; de la Higuera and Oncina 2003).

As explained in Sect. 2.1, the majority of grammars for natural language has been developed opting for context-free grammars instead of context-sensitive ones. Therefore, this survey aims at giving a comprehensive review of grammar inference methods for NL developed in the last decade and based on CFGs, classifying them according to the features highlighted in the previous section, i.e. the presentation set and the type of information. Table 1 summarizes the analysed methods that are deeply described in Sect. 3.1 and its sub-sections. The table shows that the majority of NL learning methods, proposed in the literature, is based on a text-based and unsupervised approach. The reason for that is threefold:

- First of all, unsupervised learning enables to learn larger and more complex models than supervised learning. This is because supervised learning aims at defining connections between one set of input sentences (i.e. training examples) and another set of output sentences (provided by the structured corpus). Therefore, the complexity of the learning task increases notably when learning models with deep hierarchies.

- Second, although supervised methods typically generate better results, due to the fact that they know the structure of the language for tuning their output, unsupervised methods are less time-consuming and costly because they do not require the onerous creation of the initial tree-bank of the language.
- Third, in NL it is problematic to specify all sentences that have not to be included into the grammar because learners typically get evidence about what is grammatical (correct sentences or positive examples), but no details about what is not grammatical (incorrect sentences or negative examples).

3.1 Description of the methods

This section takes an in-depth look at the grammatical inference methods for NL, summarized in Table 1.

3.1.1 ADIOS

The ADIOS (Automatic DIstillation Of Structure) algorithm (Solan et al. 2005) was proposed as a statistical method of grammar induction that yields symbolic results in the form of a context-free grammar. It induces grammars from a corpus of strings (such as text, transcribed speech, nucleotide base pairs, etc.), using only positive examples in an unsupervised fashion. Therefore, ADIOS can be classified as a *text-based* and *unsupervised* grammatical inference method.

The algorithm works in three phases: initialization, pattern distillation and generalization. Initialization involves loading the corpus onto a directed pseudograph (i.e. a non-simple graph in which both loops and multiple edges are permitted) whose vertices are all lexicon entries, augmented by two special symbols, `begin` and `end`. A sentence in the graph is represented by a path over the graph, starting at `begin` and ending at `end`, and is indexed by the order of its appearance in the corpus. In the top of Fig. 2, the pseudograph for the sentence "*that the cat is eager to please disturbs Beth*" is initialised. Initialization is followed by pattern distillation that consists in the extraction of significant patterns (i.e. sequences of nodes) from the pseudograph by finding sub-paths of high probability, considering the number of outgoing and incoming edges of a sub-path. After the candidate patterns have been generated, the generalization phase looks for finding the most significant pattern, then generalizes over the graph by creating equivalence classes from all of the variable node elements in the pattern. For instance, in Fig. 2 the equivalence class E50 consisting of the nouns {*bird, cat, cow, dog, horse, rabbit*} is created. At the end of each iteration, the most significant pattern is added to the lexicon as a new unit, the sub-paths it subsumes are merged into a new vertex, and the graph is rewired accordingly. An example of the working of the algorithm, highlighting the hierarchical construction of a context-free grammar and the resulting tree structure is shown in Fig. 2.

3.1.2 EMILE

EMILE (Adriaans 2001) was firstly proposed by Adriaans in 1992 and successively updated through the years until the latest version (Adriaans and Vervoort 2002). It is based on a teacher/pupil metaphor, where the teacher generates grammatically correct sentences (positive examples) and the pupil can ask the oracle which one is valid. Therefore, EMILE belongs to the class of *text-based* and *supervised* grammatical inference methods.

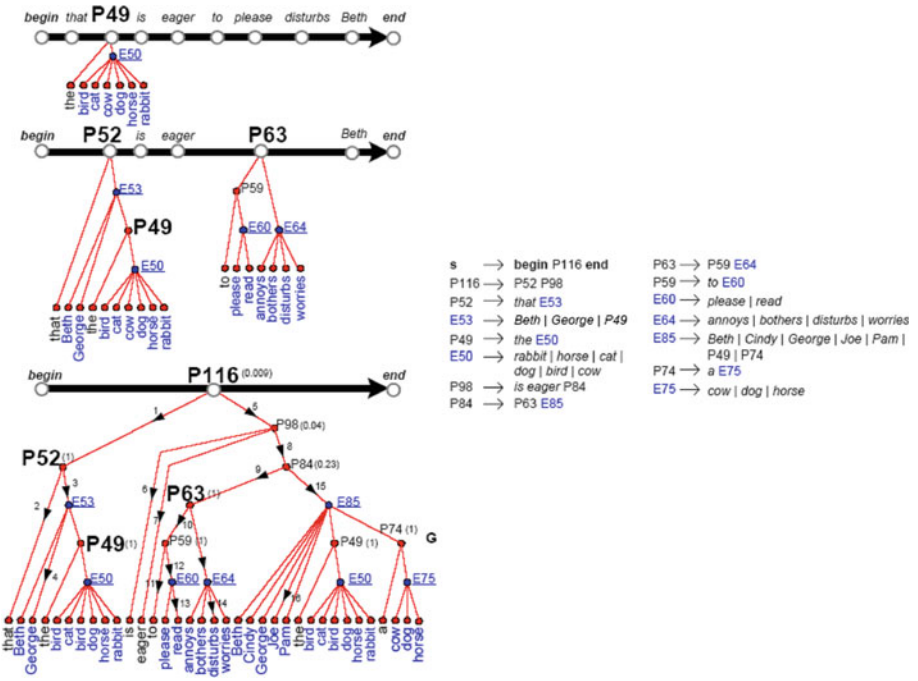

Fig. 2 Working of the ADIOS algorithm (Solan et al. 2005)

More in detail, the EMILE algorithm consists of five main steps: first order explosion, verification, clustering, rule induction and rule rewriting. Given an input set of positive example sentences, each sentence is examined to discover how it can be broken up into subexpressions. For instance, considering the example sentences "*John loves Mary*" and "*Mary walks*", a possible set of subexpressions is the following: {"*John", "John loves", "John loves Mary", "loves Mary", "Mary", "Mary walks", "walks"*}. The resulting set of subexpressions is passed to an oracle to verify substitutions of all expressions in each context. The oracle returns whether or not each subespression is a valid sentence. Afterwards, the next stage consists in clustering context rules (passed by the oracle) into types. Expressions that can be substituted into the same contexts belong to the same type. For instance, the context rules $S/lovesMary \rightarrow John$ and $S/lovesMary \rightarrow Mary$ imply that *John* and *Mary* can be substituted in the same context and therefore they belong to the same type A. In the rule induction phase, the basic and complex rules, associated to specific types during the clustering phase, are generalized toward more general types and consequently new grammar rules are introduced. For, instance, the grammar rules $S/lovesMary \rightarrow A$ and $A \rightarrow John \mid Mary$ are introduced. Finally, the rules are rewritten with the outcome of producing context-free grammar rules.

3.1.3 e-GRIDS

The e-GRIDS algorithm (Petasis et al. 2004) is a grammar inference method that is based on the GRIDS algorithm ("Grammar Induction Driven by Simplicity") (Langley and Stromsten 2000) and, like its predecessor, it utilizes a simplicity bias for inferring CFGs from positive examples only. Moreover, it does not use an oracle to check the validity of sentences. For

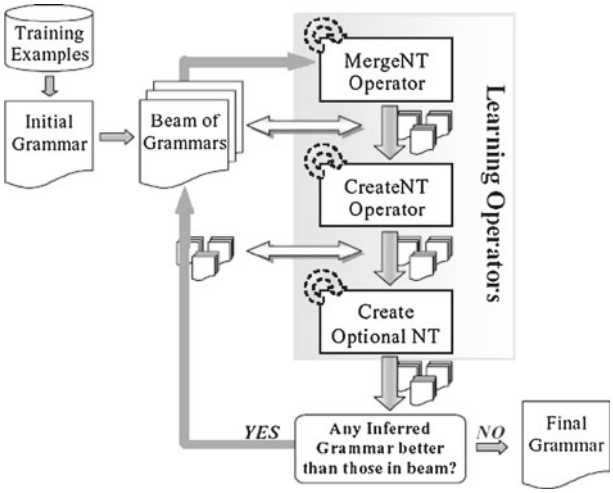

Fig. 3 The e-GRIDS algorithm (Petasis et al. 2004)

these reasons, it can be classified as a *text-based* and *unsupervised* grammatical inference method.

A general workflow of the e-GRIDS algorithm is shown in Fig. 3. e-GRIDS uses the training sentences in order to construct an initial grammar by converting each one of the training examples into a grammatical rule. Subsequently, the learning process takes place, which is organised as a beam search. Having an initial hypothesis (the initial grammar) in the beam, e-GRIDS uses three learning operators in order to explore the space of CFGs:

- *MergeNT* operator, which merges two non-terminal symbols into a single symbol X, thereby replacing all their appearances in the head and the body of rules by X;
- *CreateNT* operator, which creates a new non-terminal symbol X from two existing non-terminal symbols that are its constituent symbols.
- *Create Optional NT*, which duplicates a rule created by the CreateNT operator and appends a non-terminal symbol to the rule, thus making this symbol optional.

The learning process occurs in three steps, according to the operator that is applied. In the first step, called "merge" step, the MergeNT operator is repeatedly applied for merging non-terminal symbols in each grammar in the beam. The resulting grammars are then evaluated for deciding if replacing the grammar in the beam that has the lowest score with the newly generated grammar that has a better score. The second step is the "create" step that considers all ways of creating new terms from pairs of symbols that occur in sequence within the grammar, by repeatedly applying the CreateNT operator. Finally, in the "create optional" step all ways of duplicating a rule by the addition of an optional extra symbol at the end of the rule body are examined by repeatedly applying the CreateOptionalNT operator. The learning process terminates when it is unable to produce a successor grammar that scores better than the ones in the beam.

As mentioned above, the e-GRIDS algorithm uses a simplicity bias for directing the search through the space of CFGs and avoiding overly general grammars. This criterion measures the simplicity of a grammar through its *description length* that is defined as the sum of the number of symbols required to encode the grammar and the number of symbols required to

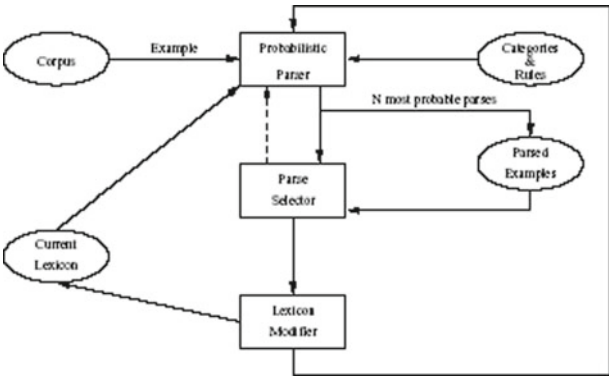

Fig. 4 A general workflow of the CLL algorithm (Watkinson and Manandhar 2001)

encode the training examples. Therefore, the algorithm directs the learning process described above towards grammars that are compact, i.e. ones that have minimum description length.

3.1.4 CLL

The CLL algorithm (Watkinson and Manandhar 2001) aims to learn natural language syntax from a corpus of declarative sentences and without the need of an oracle for validating hypotheses about the language. Therefore, it belongs to the class of *text-based* and *unsupervised* grammatical inference method.

The CLL algorithm consists of three main stages (see Fig. 4): parsing of the examples, parse selection and lexicon modification. First of all, the algorithm takes an example sentence at a time from the corpus, which is then parsed using a n-best probabilistic chart parser, developed from a standard stochastic CYK algorithm (Kasami 1965). The parsing stage results in a number of possible parses, which are sent to the parse selector for determining which one creates the most compressive lexicon. To do that, the algorithm measures the sum of the sizes of the categories for each lexical entry and evaluates the effect of the new lexicon on previous parses by re-parsing examples that may be affected. The final stage takes the current lexicon and replaces it with the most compressive lexicon selected in the previous stage. The three steps are repeated until all the example sentences of the corpus have been parsed.

3.1.5 CDC

The CDC (Context distribution clustering) algorithm was proposed by Clark (2001) for the unsupervised induction of stochastic context-free grammars from tagged text. Consequently, CDC can be classified as a *text-based* and *unsupervised* grammatical inference method.

The CDC algorithm makes use of two techniques: distributional clustering and mutual information. The former allows clustering together sets of tag sequences according to tags immediately preceding and following the tag sequence. This technique is used to identify sets of sequences that can be derived from a single non-terminal. An example of cluster is composed of the following tag sequences that can be derived from the non-terminal AT0:

{AT0 AJ0 NN0; AT0 AJ0 NN1; AT0 AJ0 NN2; AT0 AV0 AJ0 NN1;
AT0 NN0; AT0 NN1 PRP AT0 NN1; AT0 NN1} .

The latter allows evaluating the dependencies between the symbol occurring before and after the constituent (terminal or non-terminal) and it is used in the CDC algorithm to remove incorrect non-terminals.

The goal of the algorithm is to obtain the grammar with minimum *description length*. It therefore starts with a grammar with maximum description length, which has one rule for each sentence type in the corpus, and a single non-terminal. At each iteration, the algorithm clusters all frequent strings, using distributional clustering, and filter according to the mutual information criterion. It greedily selects the cluster that will give the best reduction in description length.

3.1.6 Inductive CYK

The inductive CYK algorithm was proposed by Nakamura et al. in several works (Nakamura and Ishiwata 2000; Nakamura and Matsumoto 2002; Nakamura 2003) and was implemented in an inductive grammar inference system called Synapse (Synthesis by Analyzing Positive String Examples). The algorithm synthesizes CFGs from positive and negative sample strings generating the minimum production rules, which derive positive strings, but do not derive any given negative strings. Moreover, inductive CYK does not make use of a critic. Therefore, it belongs to the class of *informant-based* and *unsupervised* grammatical inference methods.

The inductive CYK algorithm takes as input two ordered sets of positive and negative sample sentences and an initial set of production rules. For each positive sample sentence, the algorithm performs two steps that have to be repeated until the positive sentence is derived from the production rules. The first step tests whether the positive sentence can be derived from the given set of production rules by applying the CYK algorithm (Kasami 1965). During the execution of this algorithm, the set of terminal and non-terminal symbols candidates of the body of newly generated rules are tracked. The second step provides a function for adding production rules when the current set of rules does not derive the positive sample sentence. This function creates new rules starting from candidate terminal and non-terminal symbols that are produced during the first phase.

Each time the algorithm finds a rule set that derives the positive sample sentence, it checks all the negative sample sentences. The inductive CYK algorithm has non-deterministic branches, or choice points, to which the control backtracks when the process fails. Moreover, a control on the search is performed by iterative deepening on the number of rules to be generated. Starting from an initial limit of the number of rules, this limit is increased by one when the system fails to generate enough rules to parse the sample within this limit and repeats the search. If the algorithm terminates with success, it returns the set of production rules and non-terminal symbols as a result.

3.1.7 LAgts

The LAgts (Language agents) method (Briscoe 2000) relies on a computational simulation that models the behaviour of a population of language agents, which can be learners, generators and parsers. The generator (or adult) always randomly generates a positive sentence and the learner always attempts to parse and learn from it, without the aid of an oracle. For such a reason, LAgts can be classified as a *text-based* and *unsupervised* grammatical inference method.

The LAgts grammatical acquisition procedure adopts: a Generalised Categorial Grammar (GCG) with associated parameters as its underlying framework, a parser for finding the category sequences representing the input sentences, and an algorithm for updating the parameter

settings. The GCG notation represents each lexical syntactic category as a sequence of p-settings (where p denotes parameters) based on a ternary sequential encoding. For instance, the lexical categories *N* and *S* are represented by a p-setting, encoding the presence, absence or lack (*true, false* or ?, respectively) of specification of the category in the grammar. The parser uses a deterministic algorithm that operates by shifting lexical categories from an input buffer to the analysis stack, where reductions are carried out on the categories in the top two cells of the stack, if possible. When no reductions are possible, a further lexical item is shifted into the stack. When all possible shift and reduce operations have been carried out, the parser outputs either the start symbol in the top of the stack, if parsing succeeds, or a sequence of categories, if parsing fails. The algorithm for updating the parameter settings works if the parser fails. The core of this algorithm is the update function, which is applied to a sequential p-setting encoding, and returns a new setting, which is retained only if the new setting results in a successful parser.

3.1.8 GA-based

The GA-based algorithm (Sakakibara and Muramatsu 2000) inductively learns context-free grammars from partially structured examples (positive and negative), i.e. only some partial information about the grammatical structure of the given examples is available. Therefore, the GA-based algorithm belongs to the class of *informant-based* and *supervised* grammatical inference methods.

The GA-based algorithm consists of three steps:

- construction of the tabular representation of the primitive CFG for the given positive examples;
- elimination of the unnecessary non-terminals and production rules from the primitive CFG based on the given partially structured examples;
- merge of the remaining non-terminals using a genetic algorithm in order to make the grammar consistent with the given positive and negative examples.

The algorithm results in the identification of a CFG having the intended structure that is structurally equivalent to the unknown grammar.

3.1.9 ALLiS

The ALLiS (Architecture for Learning Linguistic Structures) method (Déjean 2000) generates syntactic structures from a tagged corpus and from correct (positive) examples of the language to be learned. Consequently, ALLiS can be classified as a *text-based* and *supervised* grammatical inference method.

The ALLiS method is based on theory refinement, which consists of improving an existing knowledge base so that it fits more with tagged training examples. The method is composed of two main steps. The first step builds an initial grammar composed of a set of rules that assigns to each tag of the corpus a default syntactic category, corresponding to its most frequent behaviour. The second step (refinement) compares this initial grammar with the training examples in order to identify the revision points, i.e. points that are not correctly described by the grammar. For these revision points, possible revisions are created. The best of these revisions is chosen to revise the grammar. This is repeated until no more revision points are found.

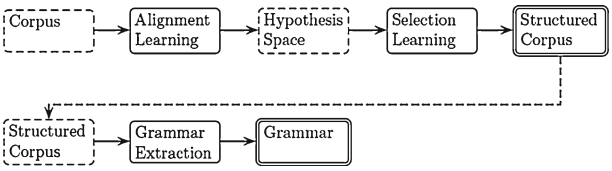

Fig. 5 A general workflow of the ABL algorithm (van Zaanen 2001)

3.1.10 ABL

The ABL (Alignment-Based Learning) algorithm (van Zaanen 2001) learns syntactic structures from positive sentences without any a priori knowledge of the language (not even part-of-speech tags of the words). Consequently, it can be classified as a *text-based* and *unsupervised* grammatical inference method.

The ABL algorithm consists of three main phases (see Fig. 5): alignment learning, selection learning, and grammar extraction phases. Alignment learning phase aligns all sentences in the input corpus such that it finds a shared and a distinct part of all pairs of sentences. Following a principle of substitutability, if two distinct parts of a sentence (called constituents) can be seen as being substitutable, they are of the same type. The second step takes the same unstructured corpus as input and tries to identify the correct constituent from the possible overlapping constituents, which are found in the previous step, by using probabilistic methods. The output of this phase is a structured tree-bank. The last phase, i.e. the grammar extraction phase, extracts a stochastic grammar from this tree-bank.

3.1.11 UnsuParse

The UnsuParse algorithm (Hänig et al. 2008) learns syntactic structures from a corpus of declarative sentences without any a priori knowledge of the language (not even part-of-speech tags of the words). Consequently, it can be classified as a *text-based* and *unsupervised* grammatical inference method.

The algorithm is based on the assumption that a word within a part of sentence (called constituent) prefers a certain position. Therefore, the first phase of the algorithm aims at computing the significance of the co-occurrence of words, by using the log-likelihood significance measure (i.e. the statistical significance of having observed a word A and then a word B n_{AB} times), and at comparing them by using a separation value that detects intuitive constituents boundaries. The algorithm proceeds by iteratively picking the smallest separation value for merging the two corresponding words into a new constituent until each sentence is a single constituent.

3.1.12 Incremental parsing

The Incremental Parsing algorithm (Seginer 2007) enables the learning of the grammar of a language from unannotated (positive) example sentences. Consequently, it can be classified as *text-based* and *unsupervised* grammatical inference method.

The algorithm is composed of two main processes: the incremental parsing and the learning processes. The incremental parsing process uses common cover links[2] for representing the syntactic structure of the sentence in input. In particular, it reads the words of the sentence one by one and adds links that have one of their ends at that word. Afterwards, the incremental parsing proceeds by calculating, at each step, a non-negative weight for every link, which may be added between the prefix and the word. The learning process takes as input a sequence of training (positive) sentences and constructs an empty lexicon. At each step, the algorithm applies the incremental parsing process for parsing one of the training sentences and updates the lexicon, accordingly, until all training sentences are parsed.

3.1.13 Self-training

The self-training method was firstly proposed by Charniak (1997) and inspired several subsequent works (McClosky et al. 2006). It learns syntactic structures from both a small set of declarative labelled examples, which are used to train an initial model, and a larger set of declarative unlabelled examples, which are labeled by the trained model and used for re-train a new model. Therefore, the self-training method belongs to the class of *text-based* and *semi-supervised* grammatical inference methods.

The self-training method proposed by Charniak (1997) applies a probabilistic model that uses a CFG for specifying how the unlabeled sentences can be parsed and which is the probability of the possible parses. Before applying this model for parsing unlabelled sentences, the parser is trained by using the small set of labelled sentences. Afterwards, the parsing of the unlabelled sentences is carried out by the trained parser applying the probability model. McClosky et al. (2006) extended the method of Charniak (1997) by introducing a further step, in which a discriminative re-ranker reorders the possible parses of each unlabelled sentence according to several features of the parses, defined in another work of the same authors (Charniak and Johnson 2005). Roughly, each feature f_j is a function that maps a parse y to a real number. The feature's value $f_j(y)$ is the number of times that the feature occurs for the parse y. For example, the feature $f_{\text{eat pizza}}(y)$ counts the number of times that a phrase in y, headed by *eat*, has a complement phrase, headed by *pizza*.

3.1.14 Co-training

The co-training method was proposed by Blum and Mitchell (1998) and successively updated by Steedman et al. (2003). It uses a small amount of manually parsed (labeled) training sentences and a larger set of unlabelled sentences (positive examples) for defining syntactic structures of a language. Therefore, it belongs to the class of *text-based* and *semi-supervised* grammatical inference methods.

Unlike self-training, the co-training method requires multiple learners, each with a different "view" of the linguistic data. In particular, the co-training method, proposed by Steedman et al. (2003), makes use of two statistical parsers, each one trained on a small set of labeled sentences. The method consists of two main phases: the scoring and selection phases. During the former, each parser assigns a label to every unlabelled sentence it parses, along with a score that estimates the reliability of the label. In the latter, these new labeled sentences are selected for being added to the two training sets in order to re-train the parsers. This procedure is iterated until the unlabeled sentences are exhausted. The method outputs, therefore,

[2] A common cover link over a sentence U is a triple (x, y, d), where $x, y \in U$, $x \neq y$ and d is a nonnegative integer. The word x is the base of the link, the word y is its head and d is the depth of the link (i.e. the maximum number of brackets between two words).

statistical parsers, whose learning capabilities are trained on the combined use of labeled and unlabeled sentences.

3.2 Analysis of the leading features of the methods

This section provides an analysis of the previously introduced grammar inference methods according to two different point-of-views: (i) the temporal evolution, and (ii) the underlying computational technique.

3.2.1 Temporal evolution

In an attempt to gain a better understanding of the current trends regarding grammar inference methods, the analysis of the temporal evolution of articles being published in the last decade and dealing with each of the methods, previously introduced in Sect. 3.1, has been carried out. The study of scientific production, based on bibliographic data, indeed, is one of the most widely used methods for obtaining indicators about temporal evolution, variations and trends in a specific field of research.

The total number of published papers about the fourteen analyzed grammar inference methods during the 11 years period 1999–2009 decreased from 10 published papers in 2000 to a minimum of 1 paper in 2009.

In particular, considering the classification of methods based on the type of information, the scientific production of supervised methods (see Fig. 6a) has been concentrated in the period from 1999 to 2004, with a peak of 5 papers in 2000. The scientific production of unsupervised methods (see Fig. 6b) grew from 1999 to 2001, reaching a peak of 7 papers in 2001. From that year up to 2007, there has been a decrease from 4 papers in 2002 and 2003 to 1 paper to 2007. In 2008, the production has shown a little increase with 3 published papers. Finally, semi-supervised methods had the most recent scientific production (see Fig. 6c), with the first publication in 2001 and the last one in 2009, reaching a peak of 3 published papers in 2003.

From the analysis of these results, we can observe that the current trend in grammar inference is oriented toward the use of unsupervised and semi-supervised approaches. The reason for that is the time-consuming and costly creation of tree-banks of languages required by supervised methods.

3.2.2 Underlying computational techniques

In this section we analyse the previously described grammatical inference approaches based on the underlying computational techniques applied to perform the language learning. In particular, the following computational techniques have been applied in the literature: statistical methods, evolutionary computing techniques, minimum description length, heuristic methods, greedy search methods, clustering techniques. Table 2 summarizes the computational techniques used in the grammar inference methods, which have been introduced in Sect. 3.1.

Grammatical inference using statistical methods consists in inferring a stochastic language, i.e. a probability distribution, in some class of probabilistic models, from empirical data, provided generally by large text corpora. Statistical techniques have been used by four of the grammar inference methods, described in Sect. 3.1 (see the first column of Table 2): Self-training (Charniak 1997; McClosky et al. 2006), co-training (Steedman et al. 2003), ADIOS (Solan et al. 2005) and UnsuParse (Hänig et al. 2008). In self-training and co-training, probabilities are assigned to all possible parses of a sentence by a statistical parser, and

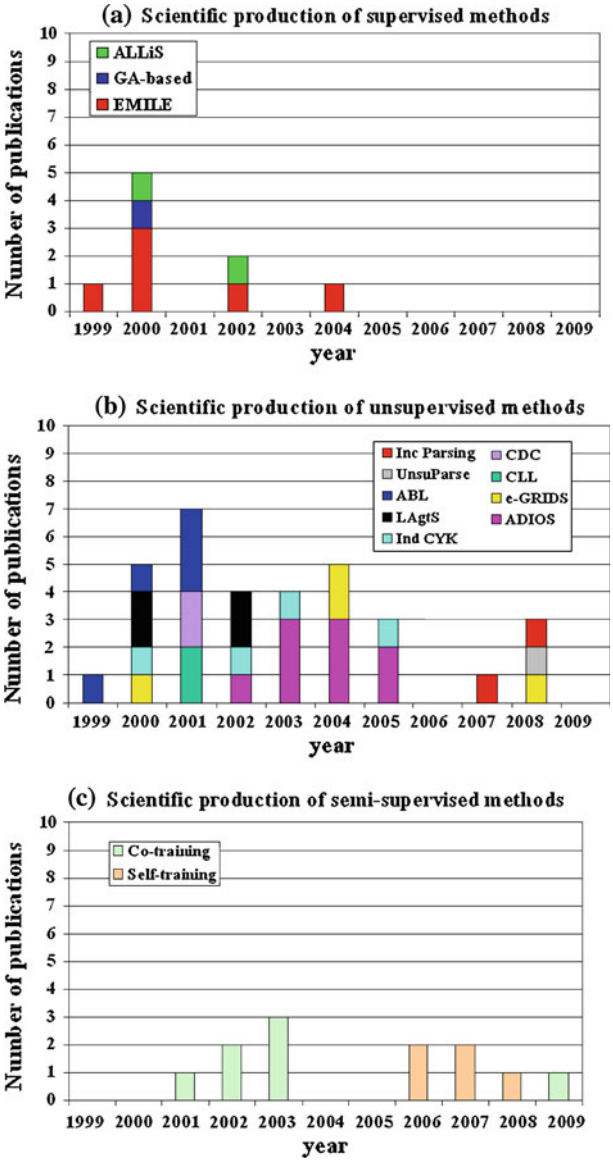

Fig. 6 Temporal evolution of the grammar inference methods

they are used for finding the high-probability parse for a sentence. In ADIOS, statistical information, present in corpus data, is used to identify significant segments of a sentence and to distil rule-like regularities. Finally, UnsuParse applies statistical methods for evaluating the co-occurrences of words.

Grammatical induction using evolutionary computing techniques consists in evolving a representation of the grammar of a target language through some evolutionary process. Two analysed methods that rely on evolutionary computing techniques are GA-based (Sakakibara and Muramatsu 2000) and LAgts (Briscoe 2000). In particular, the GA-based method uses

Table 2 Underlying computational techniques used in the analysed grammar inference methods

	Underlying computational techniques					
	Statistical methods	Evolutionary computing techniques	Minimum description Length	Heuristic methods	Greedy search methods	Clustering techniques
ADIOS	X				X	
EMILE						X
e-GRIDS			X			
CLL			X			
CDC			X		X	X
INDUCTIVE CYK				X		
LAgtS		X				
GA-based		X				
ALLis				X		
ABL				X		
UnsuParse	X					
Incremental parsing					X	
Self-training	X					
Co-training	X					

a genetic algorithm for partitioning the set of non-terminals consistently with the given examples of sentences in order to eliminate unnecessary non-terminals and production rules from the initial (primitive) grammar. LAgts applies evolutionary computing techniques in the simulation model that supports the evolution of a population of language agents.

Minimum Description Length (MDL) is a principle of statistics introduced by Rissanen (1982) that relies on the belief that the best way to capture regular features in data is to construct a model in a certain class which permits the shortest description of the data and the model itself. Grammar learning based on the MDL principle aims at constructing a grammar by means of incremental compression of the grammar rules. The MDL principle has been used by three of the grammar inference methods, described in Sect. 3.1 (see the third column of Table 2): CLL (Watkinson and Manandhar 2001), CDC (Clark 2001), and e-GRIDS (Petasis et al. 2004). In the CLL method, MDL is applied for selecting the best parse (i.e. that creates the most compressive lexicon) among the possible parses resulting from the application of a n-best probabilistic chart parser. CDC applies the MDL principle for grouping parts of production rules of an initial grammar (which has one rule for each sentence in the training set) yielding a reduction in the amount of information needed to describe the grammar. In the e-GRIDS algorithm, MDL is used for comparing grammars and selecting the one that is more "compact" in terms of the length of both the grammar and the examples of the training set.

Heuristic methods are applied in grammatical induction for exploring the space of the possible grammars, which generate the training examples of sentences, and converging the search towards the "correct" grammar. Several heuristic methods exist in the literature, according to the way the "correct" grammar is defined. The ALLiS method (Déjean 2000) applies a heuristic for reducing the number of (redundant) rules generated by the first phase of the algorithm. This heuristic consists in selecting the rules that are most frequent and having

the richest context. The ABL algorithm (van Zaanen 2001) makes use of the edit distance (Levenshtein 1965) as the heuristic for finding the longest common subsequence between two sentences and consequently denoting the unequal parts of the sentences as possible constituents. In the inductive CYK algorithm (Nakamura 2003), various heuristics are adopted for limiting the generation of rules according to their form, for avoiding the repetition of equivalent searches and for an intelligent backtracking.

Grammar inference by greedy algorithms makes, iteratively, decisions that seem to be the best at that stage, such as the making of a new or the removing of the existing rules, the choosing of the applied rule or the merging of some existing rules. Because there are several ways to define "the stage" and "the best", there are also several greedy grammar inference algorithms. In the Incremental Parsing algorithm (Seginer 2007), "the stage" is calculated on the basis of the training sentence that is learnt, while "the best" lexicon is defined based on the weights that are assigned to the links between the prefixes and the words of the sentence. The CDC method (Clark 2001) employs a greedy approach that, starting with the maximum likelihood grammar, having one rule for each sentence type in the corpus, and a single non-terminal, clusters at each iteration all frequent strings and filters according to the MDL principle. ADIOS (Solan et al. 2005) applies a greedy learning algorithm to the graph representing sentences for scanning significant patterns within the graph and selecting the best pattern to assign a new non-terminal category.

Finally, clustering techniques are applied for grammar induction with the aim of, starting from a grammar containing all sentences in the training set, clustering the syntactic units together until a satisfactory, generalized structure for the grammar is obtained. Clustering techniques have been used by two of the grammar inference methods, described in Sect. 3.1 (see the sixth column of Table 2): EMILE (Adriaans 2001), and CDC (Clark 2001). EMILE makes use of clustering of contexts and expressions. Specifically, expressions that occur in the same context are clustered together and are thus substitutable in similar contexts. CDC applies distributional clustering for creating sets of sequences, corresponding to syntactic constituents, based on the contexts they appear in, and then selecting clusters that satisfy the MDL principle.

From a combined analysis of the underlying computational techniques with the presentation set and type of information of the fourteen investigated grammar inference methods, we can observe that statistical approaches have been applied only by text-based methods, while they have not ever been applied by supervised methods. This result is coherent with the study of Denis (1998) that showed that languages learnable in a statistical query model are learnable from positive and unlabelled examples.

Moreover, the MDL principle has been followed by text-based and unsupervised methods only. This is due to the fact MDL is generally used as a principle for avoiding the over-generalisation of the grammar when no negative examples are available.

4 Evaluation of grammar inference methods

The evaluation of grammar inference algorithms is not a trivial task, and many different approaches have been proposed in the literature. Section 4.1 discusses three of the principal evaluation strategies usually applied for evaluating language learning algorithms. Next, a summary of how these evaluation techniques have been applied to the fourteen grammar inference methods, analysed in this paper, is given in Sect. 4.2, along with some results of the evaluation, obtained by the authors in their original works. The analysis of these results

will allow giving some indications of the advantages and shortcomings of each grammar inference method.

4.1 Evaluation strategies

Generally, the evaluation of a grammar inference algorithm is carried out by giving in input to the algorithm a set of unstructured data and evaluating its output. The different evaluation techniques applied in the literature can be grouped into three strategies: the "*Looks-Good-to-me*", the "*Compare Against Treebank*" and the "*Rebuilding Known Grammars*".

The "*Looks-Good-to-me*" approach has prevailed for many years due to its apparent simplicity. When a grammar inference algorithm is evaluated using this approach, the algorithm is applied to a piece of unstructured text and the resulting grammar is qualitatively evaluated on the base of the linguistic intuitions of the evaluator, that highlights the grammatical structures which look "good". As this approach needs only unstructured data as input, it can be evaluated on different languages without the need of structured corpora (van Zaanen 2001). However, the method has many disadvantages. First of all, this kind of evaluation is mainly conducted by an expert who has specific knowledge of the syntax of the language, which is generally the developer of the system. This leads to a high chance of a biased evaluation, making it almost impossible to gain an accurate measure of system performance.

Another approach for evaluating grammar inference algorithms is the "*Compare Against Treebank*". This evaluation method consists in applying the grammar inference algorithm to a set of plain natural language sentences that are extracted from an annotated treebank, which is selected as a "gold standard". The structured sentences generated by the algorithm are then compared against the original structured sentences from the treebank. A schema describing how a learning system is evaluated against a treebank is shown in Fig. 7. There are several metrics that can be used to compare the learned tree against the original tree structure. Most often, the recall, which gives a measure of the completeness of the learned grammar, and the precision, which shows how correct the learned structure is, are used. These two metrics (detailed in Sect. 4.2.2), along with the crossing brackets, which is a metric counting the number of response constituents that violate the boundaries of a constituent in the key, belong to the PARSEVAL scoring metrics, proposed by Black et al. (1991) for comparing a candidate parse (the output of the algorithm) with its reference parse from the annotated corpus. Another metric is the f_1 score, which can be interpreted as a weighted average of the precision and recall metrics. The "*Compare Against Treebank*" method does not need an expert to indicate if some construction is correct or incorrect, allowing for a relatively objective comparison of different algorithms. The main problem with this approach is that structured corpora are needed.

The "*Rebuilding Known Grammars*" approach is another evaluation method. This method, starting from a pre-defined (simple) grammar, generates a set of example sentences, which are given as input to the grammar inference algorithm and the resulting grammar is compared manually to the original grammar. If the inferred grammar is similar or equal to the original grammar then the learning system is considered good. A schema describing how a learning system is evaluated rebuilding known grammars is shown in Fig. 8. The advantages of this evaluation method are quite similar to the "*Looks-Good-to-me*" approach. An additional advantage, similarly to the "*Compare Against Treebank*" method, is that the evaluation can be done automatically, without the need for a language expert, and, therefore, it yields a more objective way of comparing different algorithms. One of the disadvantages of this approach is that the evaluation heavily depends on the chosen grammar.

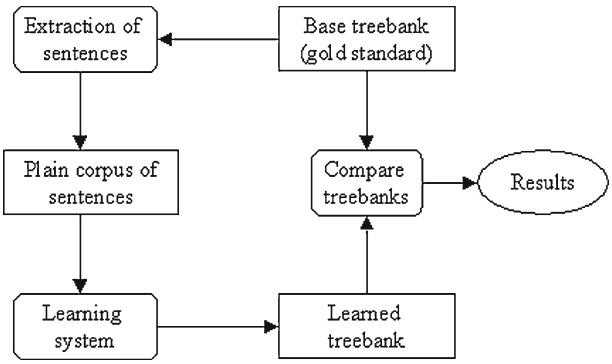

Fig. 7 Functioning of the *Compare Against Treebank* strategy

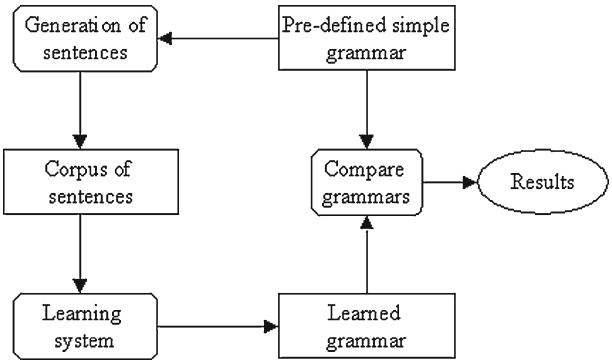

Fig. 8 Functioning of the *Rebuilding Known Grammars* strategy

4.2 Evaluating grammar inference methods

The grammar inference methods, introduced in Sect. 3.1, have been evaluated by the authors of the methods using one of the three different evaluation techniques, described in the previous section. Table 3 summarizes the evaluation strategy applied to each grammar inference method. The table shows that the majority of NL learning methods has been evaluated using the "*Compare Against Treebank*" strategy. The reason for that is twofold. First of all, this evaluation strategy is unbiased with respect to the evaluator and is scalable. Consequently, it allows obtaining objective results. Secondly, it comes closest to the evaluation of a system in a real context.

The evaluation process of each of the fourteen grammar inference methods and the obtained results are discussed in the following sub-sections.

4.2.1 Grammar inference methods evaluated through the looks good to me

The "*Looks-Good-to-me*" approach has been used to evaluate only one of the fourteen grammar inference methods, that is the LAgts algorithm. Probably, this is due to the evolutionary nature of the LAgts grammatical acquisition process that does not fit with the use of an evaluation method that relies on a pre-defined set of structured sentences, like the "*Compare Against Treebank*" and the "*Rebuilding Known Grammars*" methods.

Table 3 Evaluation strategies applied to the analysed grammar inference methods

	Evaluation strategies		
	Looks-good-to-me	Compare Against Treebank	Rebuilding Known Grammars
ADIOS		X	
EMILE		X	
e-GRIDS			X
CLL		X	
CDC		X	
INDUCTIVE CYK			X
LAgtS	X		
GA-based			X
ALLis		X	
ABL		X	
UnsuParse		X	
Incremental parsing		X	
Self-training		X	
Co-training		X	

The experimental setting for the LAgts algorithm consists in enabling a linguistic interaction between a randomly selected agent emitting a sentence and another randomly selected agent parsing it. The experiments evaluate the effectiveness of the acquisition procedures over a set of eight different languages in terms of number of input sentences required by the agents to converge on each of the languages. Results showed that the agents converged to the target grammar with less than a 1% error rate on the basis of 100 input sentences. Note that these results provide a qualitative evaluation of the method, and more specific performance is not available.

4.2.2 Grammar inference methods evaluated through the compare against treebank

The "*Compare Against Treebank*" strategy has been used to evaluate ten of the fourteen grammar inference methods, as shown in the second column of Table 3. As mentioned in Sect. 4.1, this strategy makes use of an annotated treebank that is taken as the gold standard against which to compare the learned treebank. For the evaluation of the ten grammar inference algorithms the following treebanks are used:

- the Air Traffic Information System (ATIS) treebank (Marcus et al. 1993), which is an English corpus containing 577 sentences (mostly questions and imperatives) on air traffic, with a mean sentence length of 7.5 words per sentence.
- the Openbaar Vervoer Informatie Systeem (OVIS) treebank (Bonnema et al. 1997), which is a Dutch corpus containing 10.000 sentences (mostly imperatives and answers to questions), with a mean sentence length of 3.5 words per sentence.
- the Wall Street Journal (WSJ) treebank (Marcus et al. 1993), which is an English corpus containing sentences extracted from newspaper articles with a mean sentence length of 35 words per sentence.

- the Penn treebank (Marcus et al. 1993), which is a corpus consisting of over 4.5 million words of American English. The ATIS, OVIS and WSJ treebanks are extracted from this corpus.
- the NEGRA corpus, which consists of 20.602 sentences of German newspaper text.

The following metrics have been used to compare the structure of the learned treebank against the structure of the gold standard:

- precision, which measures the number of correctly learned constituents as a percentage of the number of all learned constituents. The higher the precision, the better the algorithm is at ensuring that what has been learned is correct. Formally, it is defined as:

$$precision = \frac{\sum_{s \in Sentences} |correct\,(gold(s), learned(s))|}{\sum_{s \in Sentences} |learned(s)|}$$

- recall, which measures the number of correctly learned constituents as a percentage of the total number of correct constituents. The higher the recall, the better the algorithm is at not missing correct constituents. It is formally defined as:

$$recall = \frac{\sum_{s \in Sentences} |correct\,(gold(s), learned(s))|}{\sum_{s \in Sentences} |gold(s)|}$$

- f_1 score, which is a weighted average of precision and recall, where is assumed that these two metrics are equally important. It is formally defined as:

$$f_1 score = 2 * \frac{precision * recall}{precision + recall}$$

- crossing brackets, which measures the number of learned constituents that overlap constituents in the gold standard. The lower the number of crossing brackets, the better the algorithm is at producing a grammar whose structure coincides with that of the gold standard.

Table 4 shows the treebank(s) that the authors of the ten grammar inference algorithms have adopted for the evaluation, the size of the corpus in terms of number of sentences used for the test, the mean length of these sentences in terms of number of words, and finally the metrics that have been measured along with the obtained results. For instance, the ADIOS algorithm has been evaluated on the ATIS treebank, consisting of 10.000 sentences, whose mean length is not specified in the original work of the authors (Solan et al. 2005), and the results of the evaluation are a precision of 65.7%, a recall of 30.08% and a f_1-score of 42%.

Comparing the results of the evaluation is not a feasible task, due to the differences in the use of both the reference treebanks (gold standard) and the scoring metrics. However, some literature studies have been dedicated to compare two or more of the previous algorithms (van Zaanen and Adriaans 2001; Cramer 2007; Hänig et al. 2008; Steedman et al. 2003). van Zaanen and Adriaans (2001) provided an interesting comparative analysis between the EMILE and ABL algorithms. They computed the precision, recall, and f_1-score metrics of the algorithms on the ATIS and OVIS treebanks. Their evaluation provided f_1-scores of 41.4% (EMILE) and 61.7% (ABL) on the OVIS corpus, and 25.4% (EMILE) and 39.2% (ABL) on the ATIS corpus.

Cramer (2007) tested the EMILE, ABL and ADIOS algorithms on the Eindhoven treebank, a Dutch corpus containing 7000 sentences with an average length of approximately 20 words. The evaluation has been done according to the PARSEVAL metrics, and the results

Table 4 Results of evaluation of the grammar inference methods through the "*Compare Against Treebank*" strategy

	Treebank					Corpus size (sentences)	Mean sentences length (words)	Metric			
	WSJ	ATIS	OVIS	Penn	Negra			Precision (%)	Recall (%)	F_1 score (%)	Crossing brackets
ADIOS		X				10000	–	65.7	30.08	42	–
EMILE		X				577	7.5	51.59	16.81	25.35	–
CLL			X			10000	3.5	49.93	36.89	41.43	–
				X		13851	15	–	–	51.89	4.86
CDC		X				577	7.5	54.9	33.3	41.4	–
ALLis	X					2012	–	91.0	91.2	91.1	–
ABL		X				577	7.5	54.52	25.77	35.00	–
UnsuParse			X			10000	3.5	63.99	53.59	58.33	–
					X	2175	$\leq$10	53.5	66.6	59.3	–
Incremental parsing	X					7422	$\leq$10	75.6	76.2	75.9	–
Self-training				X		2175	$\leq$10	51	69.8	59	–
	X					2000000	$\leq$10	–	–	90	–
Co-training	X					1000	–	–	–	78.6	–

showed a recall of 17.3% (ABL), 0.5% (ADIOS) and 1.5% (EMILE), a precision of 11.7% (ABL), 2.8% (ADIOS) and 17.3% (EMILE), and a f_1-score of 14% (ABL), 0.9% (ADIOS) and 2.7% (EMILE). These results are lower than those obtained by van Zaanen and Adriaans (2001) due to the different and more complicated structure of the reference corpus used by Cramer (2007). This fact highlights the difficulties of these algorithms in deriving meaningful structures from more complex languages, such as the Dutch.

Hänig et al. (2008) evaluated the UnsuParse algorithm by comparing it with the Incremental Parsing. They tested the algorithms on two sets of sentences extracted from the NEGRA corpus: the former contains sentences with at most 10 words (NEGRA10), while the latter contains sentences with at most 40 words (NEGRA40). The results of this evaluation showed a f_1-score of 63.4% (UnsuParse) and 59% (Incremental Parsing) on the NEGRA10, and a f_1-score of 45.5% (UnsuParse) and 40.6% (Incremental Parsing) on the NEGRA40. This demonstrates that UnsuParse performs well than Incremental Parsing, and the performances of both algorithms decrease significantly for longer sentences.

Another comparative study is carried out by Steedman et al. (2003), which conducted various experiments to compare the self-training and co-training algorithms. The authors tested the algorithms on the WSJ Treebank (Sects. 2 to 21). The obtained results showed a f_1-score of 77.8% (co-training) and 75.1% (self-training) after 100 rounds of the algorithms. Therefore, co-training results in higher performances than self-training.

4.2.3 Grammar inference methods evaluated through the rebuilding known grammars

The "*Rebuilding known grammars*" approach has been used to evaluate the following three grammar inference methods: e-GRIDS, GA-based, and Inductive CYK.

Generally, the metrics used in this evaluation strategy are devoted to compare the induced grammar against the artificial "correct" grammar, evaluating whether the inferred grammar is similar or equivalent to the original grammar in terms of:

- errors of omission (failures to parse sentences generated by the "correct" grammar), which indicate that an overly specific grammar has been learned;
- errors of commission (failures of the "correct" grammar to parse sentences generated by the inferred grammar), which indicate that an overly general grammar has been learned;
- expressiveness of the inferred grammar (ability to parse correctly sentences longer than sentences used for training).

e-GRIDS was evaluated by using an artificial grammar including declarative sentences with arbitrarily long strings, and involving recursion in order to describe an infinite language. From this grammar a set of 22.826 sentences was generated, which is split in two subsets: the first containing sentences with length up to 15 words, and the second containing sentences with length between 16 and 20 words. The algorithm was tested on the sentences taken from the first and second sets. The results of the experiment showed that the inferred grammar caused a number of errors of omission equal to 0.1 (10%) if more than 350 sentences are given in input, both shorter (<15 words) and longer (16–20 words). Moreover, the inferred grammar has a probability of 15% of causing errors of commission.

Similarly to the e-GRIDS, the GA-based algorithm was tested by using an artificial grammar that is recursive. Seven partially structured examples are generated from this artificial grammar. Giving in input these examples to the GA-based learning algorithm, the number of iterations of the algorithm that are needed for converging toward the correct grammar is evaluated. The results showed that more structured examples are given to the algorithm, the lower is the number of iterations to be converged to the correct grammar.

The Inductive CYK algorithm was evaluated by computing the computation time that the algorithm employs for synthesizing a set of six artificial grammars, which are those presented in the textbook by Hopcroft and Ullman (1979). The grammars are synthesized starting from a set of positive samples with various length. The results of the evaluation showed that the computation time increases as the longer positive samples are given first. This is due to the fact that the algorithm generates several rules at one time and requires more time for testing a larger number of possible sets of rules.

The "*Rebuilding known grammars*" evaluation strategy does not provide a unique measure of the performance of the inference algorithms. The e-GRIDS, indeed, used the errors of omission and commission, the GA-based adopted the number of iterations and, finally, the Inductive CYK evaluated the computation time. This is the main reason of the incomparability of these grammar inference methods.

5 Conclusion

In this survey, we discussed the problem of natural language learning and gave an overview of the existing grammar inference methods for natural language.

We took into account the kind of presentation (if text or informant) and the type of information (if supervised, unsupervised, or semi-supervised) to broadly classify the grammar inference methods for natural language into *informant-based* and *text-based* methods, and *supervised, unsupervised,* and *semi-supervised* methods, respectively.

The introduced methods have been then analyzed considering how they have evolved in time and taking into account their underlying computational techniques.

In particular, an analysis of scientific production, based on bibliographic data, has been carried out for obtaining indicators about temporal evolution, variations and trends in the field of grammar inference methods.

The current state of the art includes lots of underlying computational techniques applied to perform the language learning: statistical methods, evolutionary computing techniques, minimum description length, heuristic methods, greedy search methods, clustering techniques. An overview of the techniques that are applied in each of the investigated grammar inference methods has been provided in the paper.

The great number of proposed grammar inference methods causes the need for systematic evaluation and comparison. Three main evaluation strategies are proposed in the literature: the "*Looks-Good-to-me*", the "*Compare Against Treebank*" and the "*Rebuilding Known Grammars*". The majority of NL learning methods has been evaluated using the "*Compare Against Treebank*" strategy, mainly due to its ability of obtaining objective results.

From this overview, the following conclusions can be drawn:

- the current trend in grammar inference is oriented toward the use of unsupervised and semi-supervised approaches. This is due mainly to the time-consuming and costly creation of tree-banks of languages required by supervised methods;
- the majority of NL learning methods, proposed in the literature, is based on an unsupervised approach. This is due to the fact that unsupervised learning enables to learn larger and more complex models than supervised learning and they are also less time-consuming and costly because they do not require the onerous creation of the initial tree-bank of the language;
- the majority of NL learning methods, proposed in the literature, is based on a text-based approach. Learners, indeed, typically get evidence about what is grammatical (positive samples), but no details about what is not grammatical (negative samples).

References

Adriaans PW (2001) Learning shallow context-free languages under simple distributions. In: Opestake A, Vermeulen K (eds) Algebras, diagrams and decisions in language, logic and computation, CSLI/CUP

Adriaans PW (1992) Language learning from a categorial perspective. PhD thesis, University of Amsterdam, Amsterdam

Adriaans PW, Vervoort M (2002) The EMILE 4.1 grammar induction toolbox. In: Adriaans P, Fernau H, van Zaanen M (eds) Grammatical inference: algorithms and applications: 6th international colloquium: ICGI 2002. Lecture notes in computer science, vol 2484. Springer, Heidelberg, pp 293–295

Angluin D (1982) Inference of reversible languages. J ACM 29:741–765

Baker JK (1979) Trainable grammars for speech recognition. In: Klatt DH, Wolf JJ (eds) Speech communication papers for the 97th meeting of the acoustical society of America, pp 547–550

Black E, Abney S, Flickinger D, Gdaniec C, Grishman R, Harrison P, Hindle D, Ingria R, Jelinek F, Klavans J, Liberman M, Marcus M, Roukos S, Santorini B, Strzalkowski T (1991) A procedure for quantitatively comparing the syntactic coverage of English grammars. In: Proceedings of the DARPA speech and natural language workshop, pp 306–311

Blum A, Mitchell T (1998) Combining labeled and unlabeled data with cotraining. In: Proceedings of the workshop on computational learning theory

Bonnema R, Bod R, Scha R (1997) A DOP model for semantic interpretation. In: ACL 1997, pp 159–167

Briscoe T (2000) Grammatical acquisition: inductive bias and coevolution of language and the language acquisition device. Language, pp 245–296

Charniak E, Johnson M (2005) Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In: Proceedings of the 43rd annual meeting of the ACL, Ann Arbor, pp 173–180

Charniak E (1997) Statistical parsing with a context-free grammar and word statistics. In: Proceedings of the fourteenth national conference on artificial intelligence, Menlo Park. AAAI Press/MIT Press

Chomsky N (1957) Syntactic Structures. The Hague Mouton.

Clark A (2001) Unsupervised induction of stochastic context-free grammars using distributional clustering. In: ConLL '01: Proceedings of the 2001 workshop on computational natural language learning, Morristown, NJ, USA. Association for Computational Linguistics, pp 1–8

Cramer B (2007) Limitations of current grammar induction algorithms. In: Proceedings of the 45th annual meeting of the ACL: student research workshop, June 25–26, 2007, Prague, Czech Republic

Déjean H (2000) ALLiS: a symbolic learning system for natural language learning. In: Cardie C, Daelemans W, N'edellec C, Tjong Kim Sang E (eds) Proceedings of the fourth conference on computational natural language learning and of the second learning language in logic workshop; Lisbon, Portugal. Held in cooperation with ICGI-2000, pp 95–98

de la Higuera C, Oncina J (2003) Identification with Probability One of Stochastic Deterministic Linear Languages. In: Proceedings of ALT 2003. Springer, Berlin, Heidelberg, pp 134–148

Denis F (1998) Pac learning from positive statistical queries. In: Proceedings of 9th international conference on algorithmic learning theory—ALT '98, Springer, pp 112–126

Edelman S, Solan Z, Horn D, Ruppin E (2005) Learning syntactic constructions from raw corpora. In: 29th Boston University conference on language development, Cascadilla Press

Emerald JD, Subramanian KG, Thomas DG (1996) Learning code regular and code linear languages. In: Proceedings of international colloquium on grammatical inference (ICGI-96), lecture notes in artificial intelligence 1147, Springer, pp 211–221

Garcia P, Vidal E (1990) Inference of K-testable languages in the strict sense and applications to syntactic pattern recognition. J IEEE Trans Pattern Anal Mach Intell 12(9):920–925

Gold EM (1967) Language identification in the limit. Inform Control 10:447–474

Hänig C, Bordag S, Quasthoff U (2008) UnsuParse: unsupervised parsing with unsupervised part of speech tagging. In: Proceedings of the sixth international language resources and evaluation (LREC 2008)

Hopcroft JE, Ullman JE (1979) Introduction to automata theory, languages, and computation. Addison-Wesley, New York

Horning JJ (1969) A study of grammatical inference. PhD thesis, Stanford University, Stanford:CA, USA

Kasami T (1965) An efficient recognition and syntax analysis algorithm for context-free languages. Science report, Air Force Cambridge Research Laboratory, Bedford

Koshiba T, Makinen E, Takada Y (1997) Inferring pure context-free languages from positive data. Technical report A-1997-14, Department of Computer Science, University of Tampere

Langley P, Stromsten S (2000) Learning context-free grammars with a simplicity bias. In: Proceedings of the eleventh European conference on machine learning (ECML 2000), lecture notes in artificial intelligence 1810, Springer, pp 220–228

Levenshtein VI (1965) Binary codes capable of correcting deletions, insertions, and reversals. Doklady Akademii Nauk SSR 163(4):845–848 (Original in Russian)
MacWhinney B (1991) The CHILDES project: tools for analyzing talk. Erlbaum, Mahwah
Marcus M, Santorini B, Marcinkiewicz M (1993) Building a large annotated corpus of English: the Penn treebank. Comput Linguist 19(2):313–330
McClosky D, Charniak E, Johnson M (2006) Effective self-training for parsing. In: Proceedings of HLT-NA-ACL 2006
Nakamura K (2003) Incremental learning of context free grammars by extended inductive cyk algorithm. In: de la Higuera C, Adriaans PW, van Zaanen M, Oncina J (eds) ECML workshop on learning contex-free grammars. Ruder Boskovic Institute, Zagreb, pp 53–64
Nakamura K, Matsumoto M (2002) Incremental learning of context free grammars. In: ICGI '02: proceedings of the 6th international colloquium on grammatical inference (London, UK), Springer, pp 174–184
Nakamura K, Ishiwata T (2000) Synthesizing context free grammars from sample strings based on inductive cyk algorithm. In: ICGI '00: proceedings of the 5th international colloquium on grammatical inference, London, UK, Springer, pp 186–195
Petasis G, Paliouras G, Karkaletsis V, Halatsis C, Spyropoulos CD (2004) e-GRIDS: computationally efficient grammatical inference from positive examples. GRAMMARS 7:69–110
Pullum GK (2003) Learnability. In: The Oxford International Encyclopaedia of Linguistics, 2nd edn. Oxford, Oxford University Press, pp 431–434
Rissanen J (1982) A universal prior for integers and estimation by minimum description length. Ann Statist 11:416–431
Roberts A, Atwell E (2002) Unsupervised grammar inference systems for natural language. Research report number 2002.20. School of Computing, University of Leeds
Sakakibara Y (1997) Recent advances of grammatical inference. Theor Comput Sci 185:15–45
Sakakibara Y, Brown M, Hughley R, Mian I, Sjolander K, Underwood R, Haussler D (1994) Stochastic context-free grammars for tRNA modeling. Nucleic Acids Res 22:5112–5120
Sakakibara Y, Muramatsu H (2000) Learning context-free grammars from partially structured examples. In: Proceedings of the 5th international colloquium on grammatical inference: algorithms and applications (ICGI), pp 229–240
Salvador I, Benedı JM (2002) RNA modeling by combining stochastic context-free grammars and n-Gram models. Int J Pattern Recogn Artif Intell 16(3):309–316
Seginer Y (2007) Fast unsupervised incremental parsing. In: Proceedings of the ACL 2007, Prague
Solan Z, Horn D, Ruppin E, Edelman S (2005) Unsupervised learning of natural languages. Proc Natl Acad Sci USA 102(33):11629–11634
Steedman M, Osborne M, Sarkar A, Clark S, Hwa R, Hockenmaier J, Ruhlen P, Baker S, Crim J (2003) Bootstrapping statistical parsers from small datasets. In: Proceedings of the annual meeting of the European chapter of the ACL, Budapest, Hungary
van Zaanen MV (2001) Bootstrapping structure into language: alignment-based learning. PhD thesis, School of Computing, University of Leeds, UK
van Zaanen M, Adriaans P (2001) Alignment-based learning versus EMILE: a comparison. In: Proceedings of the Belgian-Dutch conference on artificial intelligence (BNAIC), Amsterdam, The Netherlands
Watkinson S, Manandhar S (2001) A psychologically plausible and computationally effective approach to learning syntax. In: Proceedings of the workshop computational natural language learning (CoNLL-2001), pp 160–167
Yokomori T (1995) On polynomial-time learnability in the limit of strictly deterministic automata. J Mach Learn 19:153–179