Automatic Grammar Induction for Machine Translation

Benjamin Han
benhdj@cs.cmu.edu

Language Technologies Institute
Carnegie Mellon University

June 16, 2001

Abstract

The structural knowledge of natural languages, also known as a grammar, plays an essential role in all machine translation (MT) systems. Usually the development of such knowledge is a costly process involving skilled labor and prolonged period of time, and the result is difficult to be reused in a new application domain. It is thus desirable to automate such development processes, even only partially. In this paper I outline the key questions of incorporating grammar induction (GI) techniques into an MT framework, and provide possible solutions after surveying the representative GI techniques. Finally a highly flexible, monolingually-trained MT system is proposed as a prototype of the MT system with GI capability.

Contents

1 Introduction

1.1 Motivation

At an abstract level, natural language translation is a process of transforming the surface form of the source language into that of the target language while preserving the underlying meaning. Although the transformation process can be approached strictly at word-to-word level, it is almost impossible to achieve an adequate accuracy without taking the structures of the languages into account. In the rule-based machine translation (MT) paradigms, such as transfer-based and interlingua-based MT, the structural knowledge of languages has been painstakingly developed by skilled labor in the form of rules, over a long period of time [9, 65, 46, 40]. Not only the process is expensive, but the produced knowledge (rule set) is often limited to a particular application domain and thus is difficult to be ported to a new one [38]. If only the process of acquiring such structural knowledge can be automated, even partially, the development cost of an MT system can be substantially reduced, and the system can be more flexible and robust.

Even though the structural knowledge is encoded in numerical forms rather than in symbolic rules for corpus-based MT techniques, the automation of acquiring the structural knowledge of languages could benefit them as well, as some of the recent research works testified. In example-based MT (EBMT) works have been done to generalize pattern matching by inducing and using word clusters [12], while in statistical MT structural knowledge is proved to be useful for improving translation accuracy for structurally distant language pairs (such as English vs. German) and enhancing segmentation [71].

Despite the apparently different representations, the structural knowledge of languages plays a crucial role in all MT paradigms. The word 'grammar' is therefore used to collectively refer to such knowledge, and the term "grammar induction (GI)" is used to refer to the process of acquiring such knowledge. More specifically, a grammar of a particular language is the syntactic knowledge of that language, regardless of how it is encoded, and it is used to predict the validity of a sentence (e.g., translation model in statistical MT), analyze a sentence (e.g., semantic analysis), generate a sentence, etc. In particular the knowledge concerned here is not the *domain knowledge* or *world knowledge*, as is the case for the term "knowledge acquisition" [54], although the latter can be used to guide the acquisition of the structural knowledge, grammar, as humans always do[1].

It is worth noting that by adopting various machine learning techniques of identifying generalized patterns and forming production rules, corpus-based techniques gradually become more rule-based alike, while at the other end of the MT spectrum, namely for the rule-based approaches, the use of data-driven techniques to induce grammar rules makes them more and more similar to the corpus-based techniques. The most cost-efficient and effective MT system might be the one which is able to strike a perfect balance between the two radically different approaches.

Although it seems to be obvious that automatic GI techniques can be very useful for the development and the deployment of MT systems, the marriage of the two (especially for the rule-based MT) is still relatively new. In this paper I set out to investigate ways of combining the two, by first observing a list of key considerations, surveying the theoretical and practical aspects of GI, and then proposing ways of using GI in MT systems. The suggestions might seem somewhat unsubstantiated at times, but this is merely a reflection that little work has been done along these lines in the MT community.

[1] Later in Section 3.1 we shall see that the term "naming relation" used in [27] refers to *grammar* as defined in this sense.

1.2 Grammar Induction

Automatic *grammar induction* (GI)[2][39, 52] concerns the problem of acquiring the structural knowledge of human languages automatically from data. A GI device takes language data and domain knowledge as input, and induces the grammar of the language, which can be encoded in various representations, as the output. The knowledge can then be utilized by a *processor* to analyze or generate language data . Optionally the device can interact with a language *informant* (teacher) and receive feedback from the processor to gain more insight of the underlying language structures. The entire process is shown in Fig. 1.

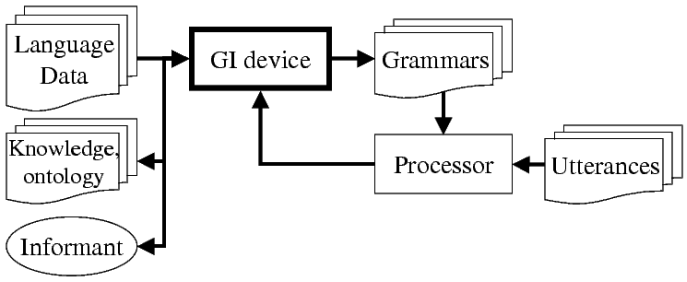


Figure 1: Grammar Induction.

In addition to the application in MT focused in this paper, GI has many other useful and important natural language applications, such as information extraction [72, 66], speech recognition [64], speech synthesis [7], etc. GI, in its broader sense, is even applied to the other problems such as DNA sequencing [58] and sequence/causality learning [62].

1.3 Organization of the Paper

The rest of the paper is organized as follows. In Section 2 a list of key considerations for combining GI and MT, motivated by both theoretical results and practical concerns, are discussed and categorized into the aspects of the input and the output of the GI component, together with the other important considerations. Section 3 then surveys the principles and practice of GI, including the notion of language learnability, and the representative algorithms for learning regular and context-free languages. With the list of key considerations in mind and the various learning techniques in our arsenal, in Section 4 we then discuss relevant issues of incorporating GI techniques in an MT framework, and suggest the idea of building an *interactive, monolingually-trained* MT system. The paper is finally concluded with Section 5, where I summarize some of the important research problems ahead toward the holy grail of building a self-adapting MT system.

2 GI-enhanced MT: Key Considerations

Before diving into the details of various GI techniques, in this section I list the key considerations for adopting a particular GI technique for MT from the aspects of the input required by a GI

[2] Also referred to as *Grammatical Inference*.

technique, the output expected for various MT subtasks, and the other practical considerations. The intersection of the answers along each dimension can then help deciding which method to choose.

The list also motivates the discussions of various theoretical and practical results presented in Section 3.

2.1 What Input Do We Need?

In considering the input required by a particular GI approach, which is categorized roughly into the symbolic and non-symbolic methods, we need to ask the following questions:

1. *How much data do we have?*
   All non-symbolic learning methods require substantial amount of data in order to converge to an adequate grammar. This requirement imposes restrictions on selecting them as the learning techniques, since collecting data incurs cost and takes time. In most of the cases preparing the training data is even more demanding than simply collecting the grammatical sentences. As will be discussed in Section 3, all of the interesting languages cannot be learned by merely observing the positive examples (grammatical sentences) without either direct or indirect negative examples (ungrammatical sentences), due to the theoretical limitations imposed by the languages themselves. Therefore additional labor is required to add more information into the collected data, e.g., giving (partial) phrase structures to the sentences, making sure the data is representative enough for the unknown language, etc., in order to make learning possible. Sometime a decision is inevitable to choose between the methods which require more but simpler data, and the techniques which need less but 'richer' data.

   To work around the problem of scarce data, one can adopt a symbolic learning technique, or choose to bootstrap the desired learning method with existing knowledge, which can be either manually developed or automatically induced. More details will be discussed in Section 4.

2. *Do we have positive examples only? Or both positive and negative examples? Any additional constraints available?*
   As discussed above GI without negative examples is not possible. If only positive examples are available, one is forced to produce whatever possible from the given data as the negative evidence. One of such methods is using the frequency of sentences as an indicator of grammaticality: if certain construct is used rarely enough to produce positive examples it will be considered ungrammatical - this is essentially learning in a probabilistic setting [61]. Some other possibilities of inducing more constraints on the pure positive examples is by exploiting domain knowledge, as I will propose in Section 4.

3. *Are the input noisy?*
   Although the symbolic methods usually require quantitatively less data and guarantee convergence, they tend to be very sensitive to any noisy input. If the goal of adopting GI is to assist experts developing an MT system, this is usually not a serious problem. However if the learning process is meant to be engaged with end users, a more sophisticated grammar/knowledge management mechanism needs to be in place to resolve any conflict between the learned grammar and the existing one. Or alternatively one can switch entirely into a non-symbolic approach.

4. *Do we need language informants? By real user interaction or by data sampling?*
   One way to obtain negative examples for an MT system is to pose questions to the user. Although theoretically an informant gives equally rich information to a GI device as does a corpus tagged with grammaticality judgments, in that both information presentation methods guarantee the learnability of most interesting formal languages [27], the use of a real human informant gives rise to several interesting possibilities, such as question planning - asking the most discriminating questions or ensuring the information garnered covers all important aspects of the unknown language, or interactive ontology construction - building domain knowledge by interacting with the end user, etc. The latter may prove useful for knowledge-based MT (KBMT).

   Alternatively an informant can be replaced by a data sampling algorithm, which operates on the given language data and gives grammaticality judgments based on the internal reasoning mechanism, thus making the GI component fully automatic.

5. *Do we need domain knowledge/ontology?*
   Although most of the conventional GI approaches use only syntactic knowledge to constrain the learning process, using the domain knowledge can be even more beneficial in making the learning problem tractable. Already such knowledge is proven to be very useful for disambiguation tasks in MT [5], however it is still not widely used in learning a grammar. In incorporating GI in MT we might want to take advantage of the existing manually developed domain knowledge/ontology, and furthermore, augment the knowledge base *online* by exploiting the power of GI. Ultimately there will be bidirectional interactions between the domain knowledge and the grammar, making the MT system highly portable across different application domains. More details will be discussed in Section 4.

2.2 What Output Do We Expect?

Given that the produced grammar will be used internally in an MT system, the following considerations are important in deciding ways of producing it in an appropriate form:

1. *What is the intrinsic complexity of the language to be learned?*
   The complexity of the language to be learned directly affects the choice of the underlying 'model' for the learning task. Conventionally natural languages are modeled by one of the *formal languages*: regular languages, context-free languages (CFL), etc, with each having its place in the continuum of the 'richness' (more in Section 3). The chosen models should be rich enough to be able to encode the desired knowledge of the language to be learned, while constrained enough so the learning problem can be made tractable.

   Although context-free grammar (CFG) is the one most frequently used to model natural languages, studies have shown that human languages are not necessarily context-free [17, 59]. On the other hand for certain subtasks in an MT system we do not actually need the full power of CFG, e.g., regular languages work just fine for morphological analysis [34] and lexicon access [41]. Even for the tasks where traditionally CFG is used, regular languages, in their equivalent forms of *finite state automata* (FSA), are shown to be able to approximate what can be done in CFG [35, 68]. Other alternatives include learning a subset of CFL, as will be discussed in Section 3.3.

In one of the classical works in statistical MT [11], Brown et al proposed five statistical translation models, with each model having an increasing complexity over the previous one. The training starts from the simplest model and uses the result to bootstrap the training of the next more complex model. This same idea of bootstrapping the learning of more powerful model using the trained simpler one can well be applied to the realm of GI, where one can learn multiple FSAs efficiently and then try to generalize upon them to induce a full CFG. However at this point I am not aware of any such attempt.

2. *What are the appropriate representations for the task?*
Almost all of the interesting formal languages have several equivalent forms. Examples include FSA and hidden Markov Model (HMM) for regular languages, and *push-down automata* (PDA), *definite clause grammars* [53], *recursive transition networks* (RTN), *augmented transition networks* (ATN) [73] and *unification-based grammars*[3] [60] for CFL. For non-symbolic GI techniques sometime a grammar is encoded in an even more different form, such as *recurrent neural networks* (RNN) [45], binary dinigt strings in *genetic algorithms* (GA) [29], and conditional probabilities in *n-gram* language models. The key concerns for picking the right representation are:

   (a) If some integration between the existing knowledge and the learned grammar is necessary, the representation should be chosen such that it can work hand in hand with the other components. E.g., other things being equal, in a rule-based MT system a GI component based on a connectionist representation would be more difficult to use since extra effort is required to extract the encoded symbolic knowledge from a neural network.

   (b) In particular if a hybrid machine/human development of the MT system is desired, the representation chosen must be human-understandable. One of the possible scenarios would be machine-aided grammar development, where human post-editing is part of the development process. Another reason for requiring a human-understandable representation is to allow the MT system the ability to *explain* the translation. The idea is to treat an MT system as a reasoning system, and interaction is possible in every step of the reasoning process in order to make the system more adaptable (more in Section 4).

   (c) Finally one should take into account whether the learned grammar will be used for analysis, generation, or transformation. Since analysis grammars are used in the first step of a complete translation process, the discussions in 2a is particularly true. On the other hand it is possible to treat the generation module in an MT system as a standalone blackbox, thus one has more freedom in choosing the representation used in the generation GI component. In Section 4.2 this idea is further developed into building an example-based generation component in order to make a monolingually-trained MT system possible.

2.3 Other Considerations

Still several questions regarding the other aspects of using GI in MT are worth pondering:

[3] Note without restrictions posed on the feature structures and unifications, a unification-based grammar becomes Turing equivalent.

1. *When does the learning take place? Online or offline?*
   Training an MT system offline means that the system can update its internal knowledge only during a certain period of time, and most likely it does this with a more controlled source of input (such as the new linguistic data collected by human experts), e.g., during the development stage of the system, or during the periodic maintenance of the system.

   If learning can take place online, i.e., the MT system or a user can initiate a learning episode anytime after the system is deployed, we can no longer have a tight control over the context of the learning processes. For one, this implies that we cannot assume that the input of the GI component is noise-free, since any end user with different level of language competence can shape the system the ways she/he likes. This consideration along requires a more sophisticated design of the knowledge management within the MT system, which is addressed in the following question.

2. *When and how is the learned grammar integrated into the existing one?*
   If the input to a GI component is noise-free, the learned grammar will always be correct[4] and can be readily merged with the existing grammar. However if the input is polluted with noise, the integrity of either the existing or the newly learned grammar is no longer certain. This uncertainty calls for a more conservative strategy in integrating the learned grammars - e.g., the integration may take place only periodically, or only if some verification criteria are satisfied. One of the other possibilities is running a controlled online verification of the new grammar, and alert the user whenever a conflict arises between the existing knowledge and the newly acquired grammar - although one must ensure that the verification can be engaged with more limited resource, as is usually the case in most online learning scenarios. More details will be discussed in Section 4.

3. *If the GI component needs to interact with language informants, what is the appropriate way for interactions?*
   Depending on a particular application domain and the expertise of the informants, the interface for interactions can range from a simple point-n-click GUI for editing complex grammatical objects (such as derivation trees, rules, state machines, etc.) [33], to a full-blown natural language dialog system which requires little even none of the linguistic expertise from the informants. In between an MT system can choose (or be instructed) to present examples of translations and awaits a user to correct them [10], or ask yes/no question for deciding proper grammatical inferences [24].

In the following section we shall turn our attention to the principles and practice of GI itself, while keeping the list of the considerations listed here in mind. Later in Section 4 we then come back to consider various issues of incorporating GI techniques into an MT system.

3 Principles and Practice of GI

This entire section is devoted to the survey of important theoretical results and representative computational realizations of GI. In particular the survey will answer the following important questions: (a) what is the mathematical conception of "learning a language"; (b) what kind of languages are

[4] But it is not necessarily optimal, in the sense that it might over-generalize/overfit over the training data.

learnable given the definition of language learnability; and (c) what are the conventional practice in learning languages? The knowledge developed here is pivotal in answering all of the key questions presented in the previous section.

In giving the mathematical rigor to the subjects so that their characteristics can be easier to understand, natural languages are usually modeled by a set of *formal languages*, which are artificial languages designed to possess different levels of *power*. Table 1 illustrates the spectrum of the formal languages. A language shown in the table properly includes the one appears in the lower row, and thus is said to be more powerful or richer. Each language can be completely specified in a corresponding grammar formalism, or alternatively can be generated/recognized by an abstract *machine*, which is a mathematical computation model. Among the languages studied the most interesting ones for our purposes are regular languages and context-free languages (CFL), although as models they are not necessarily capable of capturing *all* observed linguistic phenomena.

Language	Grammar	Machine	Chomsky Hierarchy
Non-computable			
Recursively enumerable (RE)	Unrestricted	Turing machines	0
Recursive		Turing machines that always halt	
Context-sensitive	Context-sensitive grammar	Linear-bounded automata	1
Context-free	Context-free grammar (CFG)	Non-deterministic push-down automata (NPDA)	2
Regular	Regular expressions	Finite State Automata (FSA)	3

Table 1: Power spectrum of different formal languages.

In the rest of the section we then turn our attention from learning a natural language to learning an abstract formal language so that concrete theoretical conclusions can be obtained[5]. But it is important to realize the difference between the two - we can always fine-tune the theoretical model to better match the particular natural language at hand.

3.1 Language Learnability

In his 1967 paper [27], Gold addressed two of the most important questions in learning a formal language: he defined the notion of *learnability models*, and examined the learnability of many formal languages. The results he obtained has great impact on the design and use of a particular GI technique.

A learnability model consists of the following three components:

1. *A definition of learnability*:
   This answers the question of "what do you mean by successfully learning a language?". Three definitions were suggested, and among them *language identification in the limit* was adopted

[5]Thus in the rest of this section the term 'language' is used to refer to a formal language.

for the rest of the investigation[6]. A language is identifiable in the limit if and only if there exists a learning device which is able to guess the correct language at some finite point of time, and never changes its guess from that point on no matter what additional information it receives. This definition is of interest because once a language is shown that it is not possible to be identified it in the limit under a certain learnability model, then there is no hope in learning it under the model, since the most general condition is already granted and it is still not identifiable. On the other hand, if a language can be identified in this sense, it may still not be practical to do so.

2. *A method of information presentation:*
   This concerns the way the information of the language is conveyed to a learning device. There are two major categories of presentation: *text* and *informants*. In the former only the grammatical sentences (positive examples) are presented to the learner[7], and in the latter all possible sentences accompanied by their corresponding grammaticality judgments are presented (hence the information includes both positive and negative examples).

3. *A naming relation:*
   A name of a language is a handle through which we can uniquely identify the language. One possible representation of names is via rule-based grammars, but we do not rule out the other possibilities. The relation between names and languages is thus an onto mapping, and the problem of language learning is reduced to finding a name such that it maps to the correct language. Two naming relations are considered: *tester* is for language analysis/comprehension and *generator* is for language generation/production[8].

Under the combinations of six different information presentation methods (three for each possibility) and two possible naming relations, Gold then investigated the learnability of a set of formal languages with respect to the total 12 models. The result is shown in Table 2.

Perhaps not surprisingly, with positive examples (text) the only learnable language is the one with finite cardinality, which is probably never useful in any natural language application. However in some situations learning via text is the only possibility[9], and there are at least three possible ways to work around the theoretical limitation:

1. Have more structural information added into the positive examples, or posit more constraints on the input data. The structural information could range from fully labeled parse trees to unlabeled partial bracketing strings.

2. Restrict the class of the languages to be learned. Depending on the application domain we might only need to learn a subset of CFL, for example, whose learnability is not rejected (yet).

3. Switch to a probabilistic framework. This is essentially a way to infer the indirect negative evidence from the frequency of occurrences of the training data.

[6] From this point on the term 'identification' is used interchangeably with the term 'learning'.

[7] In the original paper a text contains *all* grammatical sentences of the target language. This is reasonable given that we are concerned with the learnability *in the limit*. The same is true for informants.

[8] It is interesting to note that the learnability of an analysis grammar entails the learnability of the corresponding generation grammar, but not vice versa.

[9] With human infants learning their mother tongues being one of the 'situations' mentioned here. It is commonly believed that infants do not receive any direct negative evidence, at least a reliable one, during the learning process, yet they still manage to pick up their languages at an amazing rate [8, 30].

Learnability model	Class of languages
Anomalous text	RE
	Recursive
Informant (all 6 models)	Primitive recursive
	Context-sensitive
	Context-free
	Regular
	Superfinite (all lngs of finite card. & at least one lng of infinite card.)
Text (5/6 models)	Finite cardinality languages

Table 2: The learnability of formal languages [27].

As will be discussed in the rest of the section, all learning approaches either use a language informant (sometime in the form of having explicit negative examples), or adopt one of the strategies listed above.

3.2 Learning Regular Languages

Being the first language with infinite cardinality, regular languages are probably the easiest ones to learn, and as argued in Section 2.2 they are useful even when the ultimate goal is to learn a CFG. However it has been shown that even learning regular languages is an NP-hard problem [28]. Clever tricks thus are still necessary to restrict the search space in order to make the problem tractable.

Most of the learning approaches adopt *deterministic finite automata* (DFA) as their underlying representation, out of the other equivalent representations such as regular expressions and non-deterministic finite automata. The reason is that for DFA a unique automaton with minimal number of states exists, and there also exist polynomial-time algorithms for equivalence/subsumption tests and minimization of DFAs [31].

In the rest of this sub-section the surveyed approaches are categorized based on the information presentation method it requires (text or informant), and the characteristics of the approaches themselves (symbolic, numeric or hybrid). This is summarized in Table 3.

A recurrent theme among all of the surveyed learning approaches is the construction of the most specific DFA according to the positive examples and the use of state merging/splitting operations. This can be summarized as follows:

1. From a set of positive examples S^+ construct the most specific DFA.
2. Generalize/specialize a more specific/general DFA by considering a set of negative examples S^-.

We illustrate this with a simple example (see Fig. 2). Let $L = \{b, aa, aaaa\}$ be the target[10] language

[10]In this paper sometime the term "target language" refers to the language to be learned, instead of the language

Info presentation \ Learning method	Symbolic	Hybrid	Numeric
Text			Learning Stochastic FSA Learning Hidden Markov Models by Bayesian Model Merging
Informant	Version space Regular positive and negative inference L* algorithm	Genetic algorithms	Recurrent Neural Networks (RNN) Learning RNN with Evolutionary Programming

Table 3: Approaches for learning regular languages.

and $S^+ = L$. The most specific DFA, a *prefix tree automaton* (PTA), is first constructed trivially by adding an accepting transition sequence for each positive example in S^+. The PTA only accepts strings in $S^+$and nothing more, and it essentially hypothesizes a partitioning of the state space Q where each state forms a single partition. At the other end we can trivially construct the most general DFA - the *universal DFA* - by merging all states into one accepting state, which in effect accepts all possible strings. We then form different partitionings of the state space by merging states (bottom-up) or splitting states (top-down), until the resulting DFA, a *quotient DFA*, conforms with both S^+ and S^-.

In [19] it has been shown that as long as S^+ is a *structurally complete* set, the search space implied by the state merging/splitting operations is guaranteed to contain the correct target DFA. S^+ is structurally complete if for each transition in the target DFA there exists at least one string in S^+ that makes use of the transition, and for each accepting state there exists at least one string in S^+ that ends at the state; e.g. $S^+ = \{b, aa, aaaa\}$ in our example is (trivially) structurally complete with respect to L.

Based on this theme, in [51] an approach using *version space* is proposed. The algorithm requires a structurally complete S^+ and an informant as the input, and uses the negative examples received from the informant to reject any quotient DFA that accepts them. The time complexity is exponential in the size of the initial PTA. Another approach, *regular positive and negative inference* [48], searches through the state space by a quadratic loop over all state-merging possibilities, and rejects a quotient DFA if it accepts any string in S^-. However this approach requires that S^- prevents any non-equivalent state in the PTA from being merged. The time complexity is $O((|S^+| + |S^-|) \cdot |S^+|^2)$.

A radically different symbolic method, the *L* algorithm*, is proposed in [4]. The algorithm relies entirely on an informant, a *minimally adequate teacher*, which answers two types of questions: (a) *membership questions*: is a string in the target language? (b) *conjecture question*: is the conjecture DFA the correct one? if not a counterexample is provided. The algorithm then incrementally constructs the conjecture DFA by maintaining an observation table, until the conjecture is right

to be translated into, as used in most of the MT literature. The usage should be self-evident based on the context.

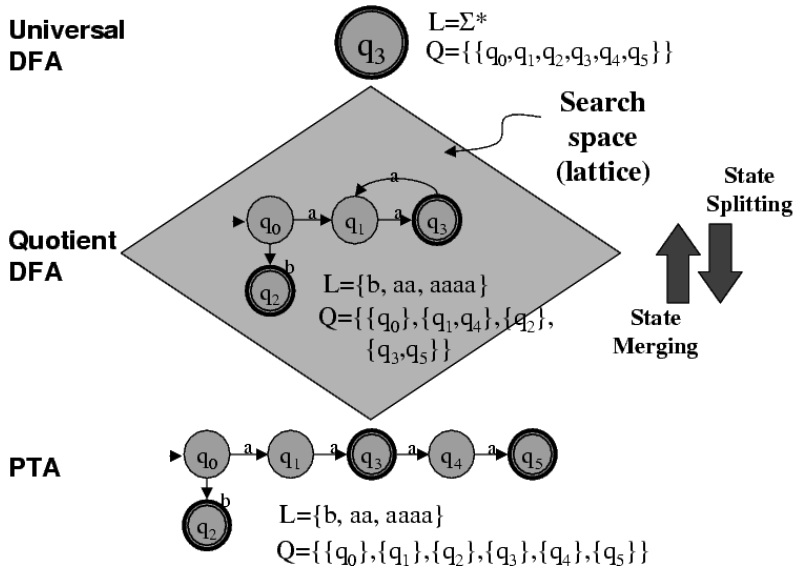
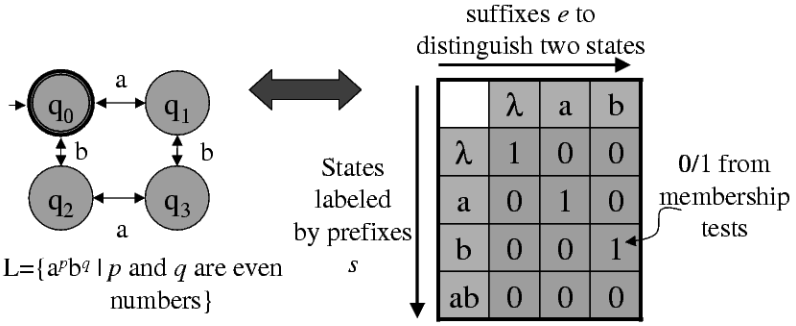


Figure 2: Searching a DFA with state merging/splitting.

on the mark. Without going into the details I will only show the correspondence between a target DFA and its observation table in Fig. 3, as a visualized motivation of the approach. In particular the membership question asked in cell (x,y) is over the string composed by concatenating the prefix label of row x with the suffix label of column y. During training only part of this table is visible to the algorithm, and it must figure out what questions to ask in order to augment the table. The algorithm is interesting in that it does not impose any restriction on the input, and an intelligent question planning might even speed up the learning process (see Section 4.1 on the role of question planning in combining GI with MT). The author further extended the algorithm into a random sampling setting for the teacher to find a counterexample should a conjecture fail. The approach can also be extended to learning a CFG.



	λ	a	b
λ	1	0	0
a	0	1	0
b	0	0	1
ab	0	0	0

Figure 3: Correspondence between a DFA and its observation table in L* algorithm [4].

The search problem can also be solved using non-symbolic methods. A possible choice is genetic algorithms (GA) [29], which has been proved effective over an ill-defined, huge search space with noise. In [19] this is done by encoding a quotient DFA in a *chromosome*. The fitness metric favors DFAs with fewer states and higher accuracy. Two GA operators are used: *structural mutation* randomly picks a state partition and randomly moves a state in the partition to another one, and *structural crossover* unions the randomly picked partitions of two parent chromosome and makes sure the resulting chromosomes are still valid.

In the connectionists camp *recurrent neural networks* (RNN) [45] have been used to model natural language phenomena because of its capability to implicitly learn a state machine. During training a set of output neurons are periodically probed to determine the state of the implicit machine, and a recurrent back-propagation method can be used to adjust the weights in order to minimize the discrepancies between the probed and the desired states. However the representation does not anticipate the possibility of an integration between the learned grammar and the existing one, which requires an additional process of extraction of symbolic knowledge from the trained network [26]. The connectionist approaches can be combined with evolutionary programming techniques to automatically find the best network topology [1].

In the statistician/Bayesian camp, works have been done in learning stochastic FSA [15], and learning stochastic grammars using *Bayesian model merging* technique [61]. The basic ideas still come from the state merging process discussed earlier, but under a probabilistic setting the merging must optimize the posterior probabilities of the training data. In the second work mentioned above, the merging takes place between non-terminals in a grammar instead of states in an abstract machine, so the approach is readily applicable to learning a CFG as well.

3.3 Learning Context-free Languages

Learning a CFG is a much more difficult task than learning a regular language: part of the reason is due to the vastly increased search space, and part of the reason is that for any two grammars G_1 and G_2, there exists no algorithm to test if $L(G_1) \subseteq L(G_2)$, or if $L(G_1) \cap L(G_2) = \phi$ - the intersection operator $\cap$ is not even closed under CFG [31]. The learning algorithms thus often rely on heuristics, or make additional assumptions on the input data or on the target language to make the problem tractable (as suggested in Section 3.1). Table 4 summarizes several representative learning techniques in a similar fashion discussed in the previous section.

Although the problem is much more challenging, several concepts from learning regular languages are still helpful in a CFG setting. First off the state merging/splitting concept is still a useful one, but it must now operate on a different representation. In [67] a learning technique based on version space is proposed. The algorithm requires both positive and negative examples ($S^+ \cup S^-$), but S^+ now contains unlabeled parse trees instead of the plain grammatical sentences. The merging happens on the non-terminals of the trees, as is the case in learning a stochastic grammar proposed in [61]. For each tree in the training data, each possible partitioning of the non-terminals results in a different grammar. The algorithm adds all possible partitionings upon seeing a positive example, and rejects a grammar and all of the grammars covered by it if it accepts any negative example. Fig. 4 shows all five possible partitionings of the unlabeled tress for utterances "*cat*" and "*black cat*". The *FastCover* operator is used to determine the partial ordering between two partitionings/grammars.

Another approach exploiting a similar idea of state-merging is proposed in [36], where a *recursive transition network* (RTN) is used as the underlying representation. As shown in Fig. 5, the

Learning method / Info presentation	Symbolic	Hybrid	Numeric
Text			Learning Stochastic CFG
Informant	RTN Learning Version space Learning reversible CFG	Genetic algorithms	Learning PDA using RNN

Table 4: Approaches for learning CFG.

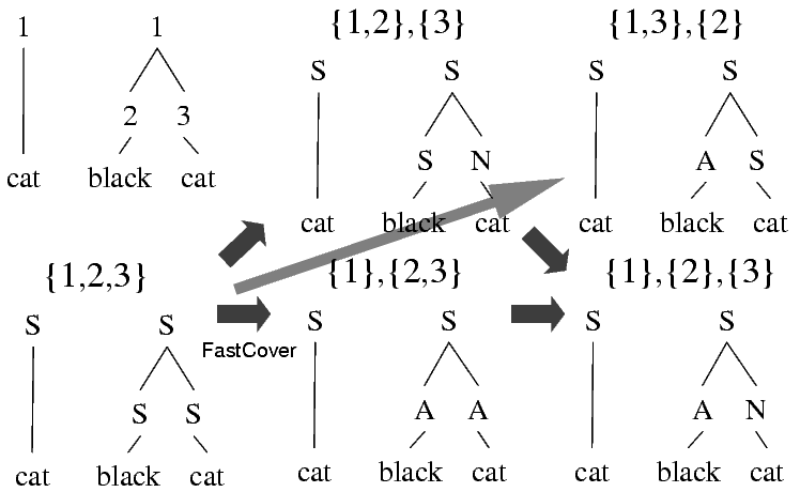


Figure 4: Version space of possible partitionings of non-terminals [67]; example taken from [52].

algorithm first constructs a trivial network covering all positive examples (very similar to the construction of a PTA described in the previous section), then it splits the network into sub-networks upon observing any significant recurring pattern. The sub-networks are then undergone a series of node mergings in order to generalize the learned grammar. It is interesting to note that the operation of splitting networks and the operation of merging nodes are actually equivalent to the *chunking* and *merging* operations proposed in [61].

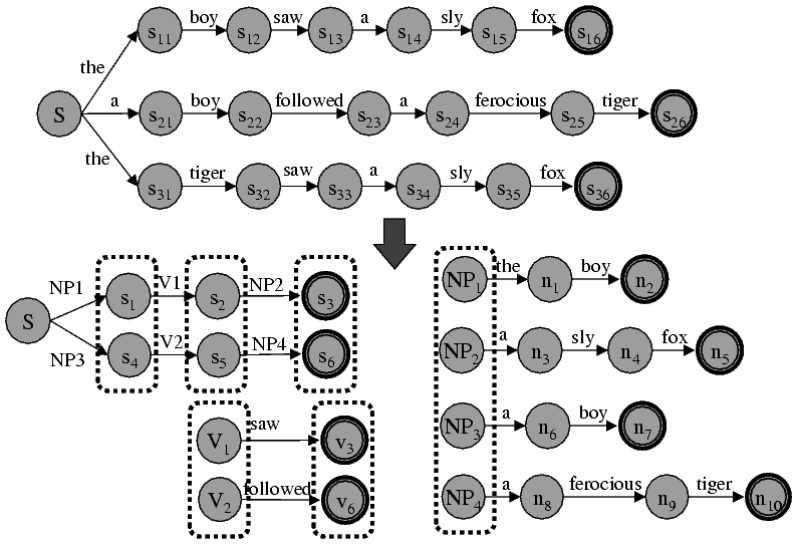


Figure 5: Learning a recursive transition network [36]; example taken from [52].

In an important work in learning CFG [56], Sakakibara proposed a normal form *reversible context-free grammar*, and showed that there exists polynomial-time learning algorithm to learn a CFG in such normal form, provided the input data contains full structural information (fully bracketing strings specified by the *parenthesis grammar* [43]). Since the algorithm operates on a normal form, it simply implies that full class of CFL can be learned if the noise-free data with structural information is available. If this precondition cannot be satisfied, one still needs to resort to the other techniques discussed in this section.

In the cases where structural information is not available, to use a symbolic technique one must make compromise in the richness of the target languages, i.e., only learn a subset of CFL. Notable examples of such subsets include *k-bounded CFG* [3], *deterministic languages* (SDL) [32], *linear* and *even linear languages* (ELL) [63], *structurally reversible languages* [13], *one-counter languages* [6], *pivot languages* [20], and *very simple languages* [74], etc.

For non-symbolic techniques, most of them bear a strong resemblance with their counterparts in learning regular languages. Obviously if a method assumes DFA as the underlying representation for learning regular languages, here it must switch to use a pushdown automaton (PDA) - a DFA with a stack. In the following works this is exactly what was proposed: in [18] an RNN is proposed to learn a PDA, and in [37] a genetic algorithm is proposed to learn a non-deterministic PDA. For probabilistic method, the use of Bayesian model merging in [61] is still valid since it operates on a

general grammar representation.

4 Incorporating GI techniques in MT

Based on the background knowledge of GI developed in Section 3, I attempt to answer all of the key questions presented in Section 2 by discussing the relevant issues in fitting GI into an MT framework. The section then concludes with a proposal of a highly flexible, monolingually-trained MT system. The discussions will focus more on the analysis (parsing) side since this is a more difficult problem, and as Section 4.2 will point out, the generation grammar can be obtained by mechanically converting from an analysis grammar, or can be discarded altogether if the MT system adopts an example-based/corpus-based generation strategy.

It should be noted that at this stage the use of GI is still not widespread within the MT community. At times the discussions in this section might seem somewhat unsubstantiated, but the intent is to hypothesize ways of combining the two at an abstract level, or at least to provoke more thinking along these lines.

4.1 Fitting GI into MT

4.1.1 Bootstrapping GI

The essential idea is to learn a grammar based on some existing knowledge in the hope that it can speed up the learning process and improve the accuracy of the learned grammar. The existing knowledge can be either syntactic, semantic (domain knowledge) or even in a completely different representation. Below I discuss four possibilities of bootstrapping: with an existing grammar, with trained simpler models, with domain knowledge, and with cross-lingual correspondences in the training data.

In the first kind of bootstrapping the GI device simply learns based on the existing grammars. This is possible provided the existing grammars have overlapping with the desired target grammar. A straightforward way of doing this, as already described in Section 3.3, is to take advantage of the vast amount of the data tagged with the correct parse trees to obtain the target grammar [16]. An indirect approach, which requires much less data, is to only augment the existing grammar whenever it is insufficient to parse certain sentences. With each unparsed sentence in the training data, the GI algorithm first reconstructs a complete parse tree (or whatever serves as the deep structure of the language) based on the incomplete parses received from the parser and the correct meaning representation tagged with the sentence, and then reads off the rules from the hypothesized tree and integrates them with the existing grammars. In [38] I have done preliminary work exactly along this line to induce *semantic grammars* on the analysis side for MT in the medical domain based on the existing grammars originally developed for the travel domain. The only difference between a semantic and a syntactic grammar is that in the former a non-terminal denotes a particular *concept* in the application domain, while in the latter a non-terminal denotes a grammatical function (e.g., nouns, verbs, etc.) [25]. An example parse tree in the semantic grammar formalism is shown in Fig. 6 and the entire GI process is summarized in Fig 7. In short the performance of the automatically learned grammar is promising compared to the manually developed one[11].

[11] The induced grammars tend to have a higher recall and a lower precision compared to the manually developed grammars.

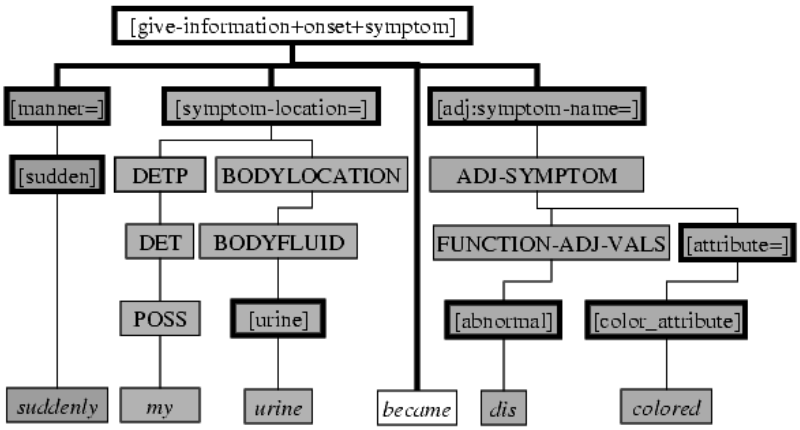


Figure 6: The parse tree of the utterance "*Suddenly my urine became discolored*" in the semantic grammar formalism.

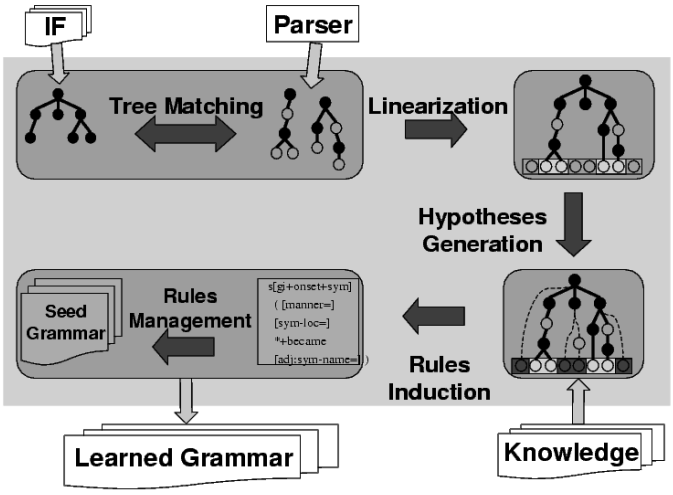


Figure 7: Outline of GI for speech-to-speech translation [38]. For each unparsed sentence the IF (Interchange Format) is tagged as its meaning representation. The algorithm then proceeds to reconstructing the correct and complete parse tree and reading off new rules from the tree.

The second possible way of bootstrapping is already motivated in Section 2.2, where a similar idea of using successively simpler models to bootstrap the training of the more complex ones, as proposed in [11], is argued to be appropriate for GI as well. For example, first a collection of simple FSAs are efficiently learned using one of the methods presented in Section 3.2 to identify only the noun phrases in both source and target language, and then a more sophisticated CFG can be learned after replacing each noun phrase with a single lexical entry representing its meaning. The same is also true for learning the transformation rules, in which the automata induced are finite state transducers instead of FSA. The grouping of simple constituents can be also achieved by using statistically motivated approaches, such as *n-gram* models.

The third possibility of bootstrapping uses the domain knowledge to constrain the GI process. Essentially this type of the approaches exploits the connections between an ontology and the grammar, and can be best illustrated when the grammar used is a semantic grammar instead of a syntactic grammar. The approach essentially consists of the following steps. Before the GI process is engaged, a skeleton grammar is first created based on the ontology. E.g., since 'breakfast' *is a* "meal type", we would expect that the full grammar should at least have a rule as shown below

$$mealType \longrightarrow \alpha\, breakfast\, \beta$$

where α and β are undetermined strings of terminals and non-terminals. The skeleton grammar can then be augmented by taking into account the input language data. Under most circumstances the ontology can even be used to constrain the generalization of the rules, such as choosing the right non-terminals to expand to for a particular left-hand side non-terminal [25]. The idea of exploiting the connections between ontologies/domain knowledge and grammars can even be extended to the point where interactive ontology augmentation is intertwined with the GI process, thus making the MT system more robust and flexible. More details are discussed in Section 4.1.3.

The last way of bootstrapping is by using the cross-lingual correspondences in the training data, and this can be best illustrated in a interlingua-based MT setting. Let s_1 and s_2 be a pair of corresponding sentences in the source language L_1 and the target language L_2, respectively, i.e., s_1 and s_2 are correct translation to each other. If the grammar for L_1 is incomplete to successfully parse s_1, we can obtain the correct meaning of s_1 by parsing s_2, provided the grammar of L_2 is sufficient to do so. With the correct meaning representation of s_1, the GI component is then able to make use of the existing grammars for L_1 to induce the missing links. This approach might find its best use in augmenting less developed grammars of some languages based on the more complete grammars developed for the other languages.

4.1.2 Introducing More Constraints on the Input

As discussed in Section 3.1, with only positive examples none of the interesting languages can be learned automatically. One way to work around the problem of having only positive examples is to posit structures or constraints on the input data, so that indirect negative evidence can be obtained. Several methods exploiting the statistics of the data (e.g., stochastic CFG), the structural information added to the data (e.g. unlabeled parse trees), etc., are already described in Section 3.2 and 3.3. Here I describe how to exploit constraints from the domain knowledge in my preliminary work of inducing semantic grammars for speech-to-speech translation as reported in [38].

Basically as described in Section 4.1.1 the particular GI technique operates on semantic grammars. The outline of the approach is shown in Fig. 7. In particular at the stage of *hypothesis*

generation, the missing links between the unparsed words and the concepts denoted by the parse nodes must be recovered. Without direct negative examples telling us what kind of pairings are not possible, such constraints must be sought out by exploiting additional knowledge. In a particular example of hypothesis generation shown in Fig. 8, one way to figure out which parse node the word '*became*' must attach to is to simply pose an ontological question to the user:

"Is the word *became* usually used to describe the onset of a symptom?"

With an answer 'yes' the correct link can then be established. The questions can either be generated using simple heuristics (e.g., if the hypothesized concept y is action-related, for an unparsed word x generate a question in the form of "is x a way to y?"), or can be canned in each of the concepts in the form of templates.

It is also possible to generate a hypothesis based on statistical evidence gathered over time. E.g., the correct link in Fig. 8 can be established if among all of the possible pairings, the pairing of word '*became*' with concept "give-information+onset+symptom" is statistically more favorable than the others given the past records.

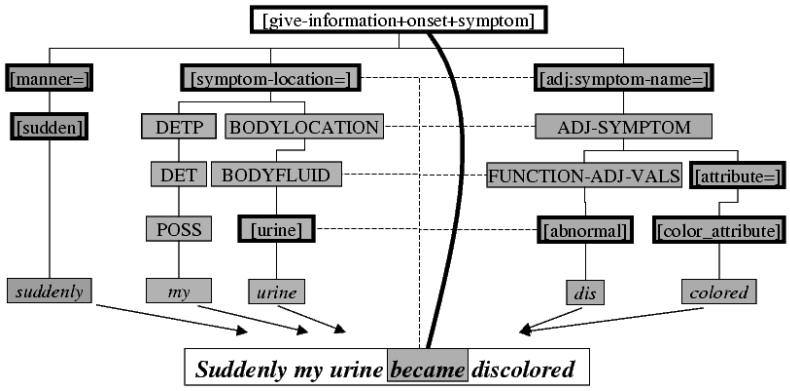


Figure 8: Hypothesis generation to pair an unparsed word '*became*' with a concept denoted by a parse node. The correct link in this case is indicated by the thick line, while the possible links are shown in dashed lines.

4.1.3 Interactive Ontology Building

As hinted in Section 4.1.1 the interactions between ontologies and grammars can be bidirectional. On the one hand an ontology can be used to bootstrap the learning process, on the other hand the flexibility brought by the GI component enables the possibility of interactively augmenting/building the ontology. As an example, imagine an MT system stumbles on a new word x. In order to figure out how to translate it correctly the system asks the user

"What is x?"

At this point the user is free to give any answer appropriate to the inquiry. Assuming our imag-

inative user is willing to cooperate by answering in a controlled language for ontology augmentation

"An x *is a* $TypeX$."

the MT system is then able to pull out the corresponding lexical category in the target language, and perhaps will ask more questions in order to zero in the correct translation. Now consider the scenario where the user feels creative and utters something like this:

"I would say an x is some kind of $TypeX$."

the MT system can then crank up its GI component and initiate further clarification questions to learn different ways of instructing ontology repairing. Thus ontologies and GI can benefit each other through a careful study of their interactions.

It is worth noting that bootstrapping the learning of the language for ontology building is intuitively simpler than bootstrapping the learning of a full natural language, since the former involves only a fixed set of concepts (e.g., *is a*, *has a*, *part of*, etc.). As a concluding remark, very little work has been done in exploring this aspect of language learning.

4.1.4 Dialog-based MT

Interactions between a user and an MT system have been exploited in several works, mainly for disambiguation at both lexical (e.g., picking the right sense of the word to translate from/to) and structural level (e.g. determining the right constituent to attach a propositional phrase) [33, 10, 22, 44]. Nevertheless, almost all of the systems with the interactive disambiguation capability do not learn or generalize the patterns acquired during such interactions[12]. With the help of GI, the interactions can not only solve an instance of an ambiguity problem, but also be able to produce useful knowledge which is general enough to resolve any future ambiguities. Furthermore, what GI is capable of is beyond mere disambiguations - novel sentences can be brought into comprehension via more carefully designed interactions.

However the keyword here is "careful design". It is noted in [44] that "interactive disambiguation leads to higher authoring costs" because of the time and efforts required by the pre-editing tasks, thus it is desirable to do away with such interactions. Despite the claim I would like to argue the contrary, namely interactions are necessary for a portable MT system, and can be made more cost-efficient via a careful design of such interactions. For the first argument, as discussed in Section 3.1, without negative examples or language informants, it is impossible to identify natural languages in the correct forms. One can claim that a complete domain model is sufficient for the purposes of negative evidence, however in practice for any non-trivial domain it is almost always the case that the domain knowledge is not complete, hence a learning mechanism is necessary to make the system robust and flexible. For the second argument, in designing the interactions between an MT system and the user, one can make them more cost-efficient by assuming less on the user's linguistic expertise, and by designing an optimal strategy for question planning. Section 4.1.1 and 4.1.3 already provide examples of hiding linguistic complexities from an end user while providing the necessary substitute for the direct negative evidence. An optimal question planning can be achieved by first defining the appropriate performance metrics (similar to the ones used in designing a spoken

[12] In system KANT [44] the decisions made in an interactive disambiguation process are recorded in the form of SGML processing instructions, but no further generalization of the pattern is done.

dialog system [69, 70]), and using various machine learning techniques such as decision trees and Bayesian networks [49, 50] for optimization. The system can even proactively plan questions without the presence of "immediate confusion", i.e., the system has the capability of self-monitoring by either periodically, stochastically, or via a reasoning mechanism, verifying the integrity of its internal knowledge, and then pose questions to remedy the predicted breakdown. Such an intelligent MT system is thus capable of, for example, probing the lexical mismatches between two languages once it senses the possibility of such mismatches, forming hypothetical questions in learning the rules of disambiguating certain linguistic constructs, or asking clarification questions after detecting a possible grammar breakdown by analogy reasoning over the past record of grammar augmentation, even before such confusion crops up in the actual use of the system.

4.1.5 Self-explaining MT

In the spirit of dialog-based MT, an ideal GI-enhanced MT system should also be able to explain to a critical user about the reasons behind a particular translation. The capability of explaining the translation decisions could help the user, regardless of a novice one or an expert MT developer, to pinpoint the problem whenever the translation is not satisfactory. For example, if the analysis grammar of an MT system cannot parse the sentence

S_1: "He runs for President."

but it can parse the sentence

S_2: "He runs for his life."

In choosing the translation for the word 'run' in the target language, which has different words for the sense of "campaigning for a position" (used in S_1) and for the sense of "physically moving one's body fast by one's feet" (used in S_2)[13], the MT system chooses the latter as the translation for sentence S_1, since the GI component decides that S_2 is the most similar analogy it can draw from in order to parse S_1 without a sufficient grammar. Upset by the seemingly counter-intuitive translation, the user demands an explanation from the system, and then realizes that a lexical mismatch causes the mistranslation. At this point the user can then proceed to add in a new lexical entry for the sense used in S_1. Without an intuitive explanatory device, this diagnostic process can range from impossible to unpleasant.

There are at least two implications for a self-explaining MT system. The first one is the problem of optimizing the explanation. The system should only report the key decisions made in learning/augmenting the insufficient grammar without boring the user to death with unnecessary details. The second one is that the underlying representation chosen for the GI processes must be suitable for high-level reasoning and explanation.

4.1.6 Knowledge Integration/Management

With the introduction of GI into an MT system the knowledge integration and management becomes an even more important problem. Ideally the integration of the existing grammar and the newly learned one needs to meet the following criteria:

[13] In Chinese these two senses are indeed realized by two different words.

1. *Correctness*: Any perturbation of the existing grammar should not alter the grammaticality judgment and the parses of the seen sentences, i.e., those grammatical sentences should still give the same parses while the non-grammatical ones should stay rejected. It is worth noting that if compactness of a grammar is preserved during the integration, over the training sessions the grammar is actually undergone a series of generalizations, in the sense that the grammar coverage is monotonically increasing. It is obvious that such generalization would cause admissions of previously ungrammatical sentences, but why is it possible to reject the existing parsable sentences? It is possible since the generalized grammar usually introduces ambiguities to the grammatical sentences. If the parser implementation only takes the top k parses into account, the truth parse might very well be pushed "beyond the horizon".

2. *Compactness*: At any time the grammar should impose least number of constraints on grammaticality such that it conforms with the given positive and negative examples.

3. *Efficiency*: The integration process, while observing the previous two conditions, should terminate in a finite amount of time. In practice we may even want to have an integration algorithm that runs in polynomial time.

First I shall discuss the situations under a symbolic setting without noise. For correctness checking, as mentioned earlier, for CFG in theory there exists no algorithm for determining if G_1 subsumes G_2. A possible way out is by keeping a *characteristic set* [2] of sentences of the target grammar as a test suite, and verify the integrity of the combined grammar over this set each time after the integration. In [42] it has been shown that for languages of *very simple deterministic pushdown automata*, the subsumption test of $L(M_1) \subseteq L(M_2)$ is decidable with exponential time complexity, by testing if $R \subseteq L(M_2)$ holds, where R is a finite characteristic set of M_1. The question is then how to restrict the grammar formalism to allow such check, and whether the restriction is reasonable given the particular application domain. For compactness checking, one of the straightforward ways is to remove any unused rule with respect to the training data. Finally for the efficiency consideration we might need to impose more restrictions on the characteristic set to make the checking more tractable, but any of such restrictions obviously voids the theoretical guarantee.

For the situations where noise is possible in the input of the GI component, one possible solution is to reduce the problem into that of *diagnosis*, where we try to pinpoint the faulty sentence, i.e., the sentence which was given with the inaccurate linguistic judgment, and asks the user to reconsider the validity of the information given before. We can make use of one of the *non-monotonic reasoning* techniques in AI to pinpoint the faulty sentence [21]. Once the conflicts between the old and the new knowledge is resolved, we can then proceed as discussed above.

In a non-symbolic setting, the correctness criterion can be recast into one that requires maximization of posterior probability of the training data under a probabilistic framework, or minimization of the error rates under a connectionists/evolutionary setting. The compactness criterion can be reduced to the requirement that the combined grammar has the *minimal description length* [55]. Finally the efficiency can be improved by pruning the redundant or insignificant training sentences (this could also be applied in the symbolic setting).

Finally it is worth noting that the knowledge integration process can be mixed with the other system activities. Already we discussed the possibility of engaging the interactive diagnostic process to pinpoint the knowledge conflicts. It is probably also the best time for the system to find out any potential confusion and initiate a proactive probing as discussed in Section 4.1.4.

4.2 Toward a Monolingually-trained MT

With all of the discussions given in Section 4.1, in this section I propose an interlingua-based and GI-enhanced MT system. Thanks to the learning capability brought by the GI component, together with an automatic generation and a dialog planning component, the system would be capable of learning to understand and speak different languages by interacting with monolinguals. The reason for choosing a symbolic approach over a non-symbolic one is that the underlying representation facilitates integration of the learned grammars with the human developed ones, and it is more straightforward to make it possible for the system to engage in a conversation with humans.

To understand what a monolingually-trained MT system can buy us, let us first consider the development process of the conventional rule-based MT systems. For a transfer-based system, each language will need its own analysis and generation grammars, and both grammars are tightly coupled with the transformation rules in between. This implies that the system requires bilingual developers, since without intimate knowledge of both source and target language it is impossible to create the necessary grammars. If a bidirectional system is desired the development effort is simply doubled. For an interlingua-based system, the source and the target languages are decoupled by the use of the interlingua - a language neutral meaning representation, thus the system only needs monolingual developers for each language. However for each language the development effort is still doubled in that the development of analysis and generation grammars are separate. Fig. 9 summarizes the discussions.

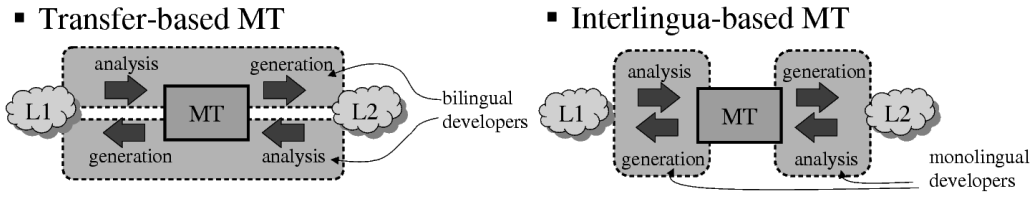


Figure 9: Development of transfer-based and interlingua-based MT systems.

Now consider the proposed monolingually-trained MT system. Since it is an interlingua-based system, it again only needs monolinguals for each language. What is more interesting is by introducing an interactive GI component using a *bidirectional grammar*, or combining an *example-based* or *corpus-based generation module*, there is no need for any manual development effort on both analysis and generation grammars. The system is then able to pick up each language by simply interacting with monolingual *speakers* - they do not even need to have sophisticated linguistic expertise. This is shown in Fig. 10.

Obviously the key to realize such highly flexible and rapidly deployable systems is the interactive GI component. All of the techniques proposed in Section 4.1: bootstrapping GI, introducing more constraints via interactions, interactive ontology building, dialog-based and self-explaining MT, and knowledge integration, can be incorporated into this single GI module. However we are still left with the question of how to actually turn an analysis grammar into a generation one.

The first possibility is to mechanically convert the learned analysis grammar into a form (not necessarily grammars) suitable for generation purposes. In [23] this is done by *inverting* an enhanced unification-based grammar used for analysis into a logical form within the framework of *Typed Feature Structures* [14]. The inversion algorithm has minor restrictions on what grammars are

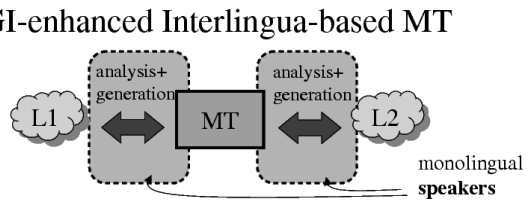


Figure 10: A monolingually-trained MT system.

invertible [57], and prior to the inversion a special-purpose feature is added to each rule constituent to keep the original phrase structure. The inverted grammar is then compiled into Abstract Machine instructions for efficient generation of surface forms. The overall framework thus provides both chart-based parsing and generation using one single grammar.

An alternative approach for generation using an analysis grammar is to bypass the creation of a generation grammar altogether. This can be done by using purely corpus-based techniques, such as the good-old n-gram model [47]. The approach has several advantages over the rule-based ones, such as extremely fast generation and better domain portability. However it suffers from its extremely naive modeling of natural languages - generated utterances can be ungrammatical or sometime even repeating part of itself. The data collection is also a problem for large application domains. Another possible solution is to make use of the learned analysis grammar, but in the reverse processing direction. As an example, consider the following meaning representation for an intended utterance "*Suddenly my urine became discolored*" (see Fig. 6; the nodes with thick borders are significant - the labels of the other nodes are not taken as the meaning of the sentence).:

give-information+onset+symptom (manner=sudden, symptom-location=urine, symptom-name=(abnormal, attribute=color_attribute))

Assuming again we are using a semantic grammar formalism, the generation can be proceeded in a *bottom-up* fashion: first the surface form of the argument-value pair "attribute=color_attribute" is generated, then the pair "symptom-location=urine", "manner=sudden" and finally the root concept, which is a dialog-act in this particular formalism, is generated. The lexical selection can be determined by maximizing the conditional probability $P(w|c)$, where w is a possible word choice and c is the underlying concept, or by maximizing over a more context-dependent class-based n-gram model. Note that the grammaticality of the generation would still be a problem, which might be remedied by a post-editing process to ensure the generation satisfies a set of local grammatical constraints.

In summary the proposed monolingually-trained MT system is able to adapt to monolinguals of different languages, and requires less development effort before it can be deployed.

5 Conclusions

In this paper I have investigated ways of incorporating GI into an MT framework, by first observing a list of key considerations, surveying the theoretical and practical aspects of GI, and then proposing ways of using GI in MT systems. In reviewing what has learned, I summarize several key research

problems as the future works toward a highly flexible and self-adapting MT system:

1. Investigating the interactions between grammar formalisms and learning techniques:
   Data scarceness is the problem plaguing almost all MT development works. With only little, purely positive data, the only way to make the learning possible is by imposing constraints on the language to be learned. A careful investigation, with a particular application in mind, of the interactions between the adopted grammar formalism and the learning technique, is thus crucial to better predict the system performance.

2. Investigating the interactions between language learning and knowledge acquisition:
   As discussed in Section 4.1.3 there seems to be an interesting way to exploit the bidirectional connections between a grammar and the domain knowledge/ontology. If used right for a knowledge-based MT system both of the costs to develop a grammar and to develop a domain model can be drastically reduced. In order to do this a more rigorous study of the relations between the two is required.

3. Investigating the interactions between the analysis grammars and generation grammars:
   It is argued in Section 4.2 that without the burden of developing two sets of grammars a GI-enhanced MT system can simply learn by interacting with monolinguals with no linguistic expertise. The key is to exploit the symmetries and understand the asymmetries between the two, and to come up with a satisfactory solution with respect to a particular application.

References

[1] P. J. Angeline, G. M. Saunders, and J. B. Pollack. An Evolutionary Algorithm that Constructs Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 5:54–65, 1994.

[2] D. Angluin. Inference of Reversible Languages. *Journal of ACM*, 29:741–765, 1982.

[3] D. Angluin. Learning k-bounded Context-free Grammars. Technical report rr-557, Yale University, 1987.

[4] D. Angluin. Learning Regular Sets from Queries and Counterexamples. *Information and Computation*, 75:87–106, 1987.

[5] K. L. Baker, A. M. Franz, P. W. Jordan, T. Mitamura, and E. H. Nyberg. Coping With Ambiguity in a Large-Scale Machine Translation System. In *Proceedings of COLING-94*, 1994.

[6] P. Berman and R. Roos. Learning One-counter Languages in Polynomial Time. In *Proceedings of the 28th FOCS*, pages 61–67, 1987.

[7] L. Blin and L. Miclet. Generating Synthetic Speech Prosody with Lazy Learning in Tree Structures. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

[8] J. N. Bohannon III and L. Stanowicz. The Issue of Negative Evidence: Adult Responses to Children's Language Errors. *Developmental Psychology*, 24(5):684–689, 1988.

[9] C. Boitet. Pros and Cons of the Pivot and Transfer Approaches in Multilingual Machine Translation. In K. Schubert and T. Witkam, editors, *New Directions in Machine Translation: Conference Proceedings*, Budapest, August 1988.

[10] C. Boitet and H. Blanchon. Multilingual Dialogue-Based MT for monolingual authors: the LIDIA project and a first mockup. in Machine Translation. *Machine Translation*, 9(2):99–132, 1995.

[11] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[12] R. Brown. Automated Generalization of Translation Examples. In *Proceedings of the Eighteenth International Conference on Computational Linguistics (COLING-2000)*, Saarbrücken, Germany, August 2000.

[13] A. Burago. Learning Structurally Reversible Context-free Grammars from Queries and Counterexamples in Polynomial Time. In *Proceedings of the 7th COLT*, pages 140–146, 1994.

[14] B. Carpenter. *The Logic of Typed Feature Structures, with Applications to Unification Grammars, Logic Programs and Constraint Resolution.* Cambridge University Press, 1992.

[15] R. C. Carrasco and J. Oncina. Learning Stochastic Regular Grammar by Means of a State Merging Method. In R. C. Carrasco and J. Oncina, editors, *Proceedings of the 2nd International Colloquium on Grammatical Inference (ICGI-1994)*, pages 139–152, 1994.

[16] E. Charniak. Tree-bank grammars. *Proceedings of AAAI-96*, 1996.

[17] C. Culy. The Complexity of the Vocabulary of Bambara. *Linguistics and Philosophy*, 8:345–351, 1985.

[18] S. Das, C. Giles, and G. Z. Sun. Learning Context Free Grammars: Capabilities and Limitations of a Recurrent Neural Network with an External Stack Memory. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 791–795, 1992.

[19] P. Dupont, L. Miclet, and E. Vidal. What is the Search Space of the Regular Inference? In R. C. Carrasco and J. Oncina, editors, *Proceedings of the 2nd International Colloquium on Grammatical Inference (ICGI-1994)*, pages 25–37, Alicante, Spain, 1994.

[20] J. Feldman, J. Gips, J. J. Horning, and S. Reder. Grammatical Inference and Complexity. Technical report cs-125, Computer Science Department, Stanford University, 1969.

[21] K. D. Forbus and J. de Kleer. *Building Problem Solvers*. MIT Press, 1995.

[22] R. Frederking, A. Rudnicky, and C. Hogan. Interactive Speech Translation in the DIPLOMAT Project. In *Proceedings of the 35th Meeting of the Association for Computational Linguistics (ACL-97)*, Madrid, Spain, 1997.

[23] E. Gabrilovich, N. Francez, and S. Wintner. Natural Language Generation with Abstract Machine. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 276–279, Ontario, Canada, August 1998.

[24] M. Gavaldà. Epiphenomenal Grammar Acquisition with GSG. In *Proceedings of the Workshop on Conversational Systems of the 6th Conference on Applied Natural Language Processing and the 1st Conference of the North American Chapter of the Association for Computational Linguistics (ANLP/NAACL-2000)*, Seattle, U.S.A, May 2000.

[25] M. Gavaldà. *Growing Semantic Grammars*. PhD thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, August 2000.

[26] C. Giles, C. Miller, D. Chen, H. Chen, G. Sun, and Y. Lee. Extraction, Insertion and Refinement of Symbolic Rules in Dynamically-driven Recurrent Neural Networks. *Connection Science - special issue on Architectures for Integrating Symbolic and Neural Processes*, 5(3,4):307–337, 1993.

[27] E. M. Gold. Language Identification in the Limit. *Information and Control*, 10(5), 1967.

[28] E. M. Gold. Complexity of Automaton Identification from Given Data. *Information and Control*, 37:302–320, 1978.

[29] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[30] P. Gordon. Learnability and Feedback. *Developmental Psychology*, 26(2):217–220, 1990.

[31] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[32] H. Ishizaka. Polynomial Time Learnability of Simple Deterministic Languages. *Machine Learning*, 2(2):151–164, 1990.

[33] D. Jones and J. Tsujii. Interactive High-quality Machine Translation for Monolinguals. In *Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-90)*, pages 43–55, Austin, TX, June 1990.

[34] K. Koskenniemi. Two-level morphology: a general computational model for word-form recognition and production. Publication no. 11, University of Helsinki: Department of General Linguistics, 1983.

[35] K. Koskenniemi. Finite-state parsing and disambiguation. In H. Karlgren, editor, *Proceedings of the 13th International Conference on Computational Linguistics*, volume 2, pages 229–232, Helsinki, 1990.

[36] P. Langley. Simplicity and Representation Change in Grammar Induction. Robotics Laboratory, Stanford University (unpublished manuscript), 1994.

[37] M. Lankhorst. A Genetic Algorithm for Induction of Nondeterministic Pushdown Automata. Cs-r 9502, University of Groningen, The Netherlands, 1995.

[38] A. Lavie, L. Levin, T. Schultz, C. Langley, B. Han, A. Tribble, D. Gates, D. Wallace, and K. Peterson. Domain Portability in Speech-to-speech Translation. In *Proceedings of the First International Conference on Human Language Technology Research (HLT-2001)*, San Diego, CA, March 2001.

[39] L. Lee. Learning of Context-free Languages: A Survey of the Literature. Technical Report TR-12-96, Center for Research in Computing Technology, Harvard University, Cambridge, MA, 1996.

[40] L. Levin, A. Lavie, M. Woszczyna, D. Gates, M. Gavalda, D. Koll, and A. Waibel. The JANUS-III Translation System. *Machine Translation (to appear).*

[41] C. L. Lucchesi and T. Kowaltowski. Applications of finite automata representing large vocabularies. *Software-Practice and Experience*, 23(1):15–30, 1993.

[42] E. Mäkinen. On the Inclusion Problem for Very Simple Deterministic Pushdown Automata. Report a-1999-8, Department of Computer Science, University of Tampere, 1999.

[43] R. McNaughton. Parenthesis Grammars. *Journal of ACM*, 14:490–500, 1967.

[44] T. Mitamura, E. Nyberg, E. Torrejon, and R. Igo. Multiple Strategies for Automatic Disambiguation in Technical Translation. In *Proceedings of the Eighth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-99)*, Chester, UK, August 1999.

[45] H. Moisl. Connectionist Finite State Natural Language Processing. *Connection Science*, 4(2):67–91, 1992.

[46] E. Nyberg and T. Mitamura. The KANT System: Fast, Accurate, High-Quality Translation in Practical Domains. In *Proceedings of COLING-92*, 1992.

[47] A. H. Oh and A. Rudnicky. Stochastic Language Generation for Spoken Dialogue Systems. In *ANLP/NAACL 2000 Workshop on Conversational Systems*, May 2000.

[48] J. Oncina and P. Garcia. Inferring Regular Languages in Polunomial Update Time. In N. Perez, editor, *Pattern Recognition and Image Analysis*, pages 49–61. World Scientific, 1992.

[49] T. Paek and E. Horvitz. Conversation as Action Under Uncertainty. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, Stanford, CA, June 2000.

[50] T. Paek and E. Horvitz. DeepListener: Harnessing Expected Utility to Guide Clarification Dialog in Spoken Language Systems. In *Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP-2000)*, Beijing, China, October 2000.

[51] R. Parekh and V. Honavar. An Incremental Interactive Algorithm for Regular Grammar Inference. In L. Miclet and C. Higuera, editors, *Proceedings of the 3rd International Colloquium on Grammatical Inference (ICGI-1996)*, Montpellier, France, 1996.

[52] R. Parekh and V. Honavar. Grammar Inference, Automata Induction, and Language Acquisition. In R. Dale, H. Moisl, and H. Somers, editors, *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text.* New York: Marcel Dekker, 2000.

[53] F. Pereira and D. Warren. Definite Clause Grammars for Natural Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence*, 13:231–278, 1980.

[54] S. D. Richardson, W. B. Dolan, and L. Vanderwende. MindNet: acquiring and structuring semantic information from text. In *Proceedings of COLING '98*, Montréal, Canada, August 1998.

[55] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific, New Jersey, 1983.

[56] Y. Sakakibara. Efficient Learning of Context-free Grammars from Positive Structural Examples. *Information and Computation*, 97, 1992.

[57] C. Samuelsson. An Efficient Algorithm for Surface Generation. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1414–1419, Montreal, Canada, August 1995. Morgan Kaufmann.

[58] D. B. Searls. The Computational Linguistics of Biological Sequences. In L. Hunter, editor, *Artificial Intelligence and Molecular Biology*, pages 47–120. AAAI Press, 1992.

[59] S. Shieber. Evidence Against the Context-freeness of Natural Languages. *Linguistics and Philosophy*, 8:333–343, 1985.

[60] S. M. Shieber. *An Introduction to Unification-based Approaches to Grammar.* CSLI, Stanford, 1988.

[61] A. Stolcke and S. Omohundro. Inducing Probablistic Grammars by Bayesian Model Merging. In R. C. Carrasco and J. Oncina, editors, *Proceedings of the 2nd International Colloquium on Grammatical Inference (ICGI-1994)*, Alicante, Spain, September 1994. Springer-Verlag.

[62] R. Sun and L. Giles, editors. *Sequence Learning: Paradigms, Algorithms, and Applications.* Springer, 1998.

[63] Takada. Grammatical Inference for Even Linear Languages. *IPL*, 2(4):193–199, 1988.

[64] F. Thollard. Use of grammatical inference in natural speech recognition. Technical report, EURISE, Saint Etienne University, 1998.

[65] G. Thurmair. Complex Lexical Transfer in METAL. In *Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Austin, TX, 1990.

[66] J. Turmo and H. Rodríguez. Learning IE Rules for a Set of Related Concepts. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, September 2000.

[67] K. Vanlehn and W. Ball. A Version Space Approach to Learning Context-free Grammars. *Machine Learning*, 2:39–74, 1987.

[68] A. Voutilainen and P. Tapanainen. Ambiguity resolution in a reductionist parser. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht University, The Netherlands, 1993.

[69] M. A. Walker, C. A. Kamm, and D. J. Litman. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering: Special Issue on Best Practice in Spoken Dialogue Systems.*, 2000.

[70] M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella. Evaluating Spoken Dialogue Agents with PARADISE: Two Case Studies. *Computer Speech and Language*, 12(3), 1998.

[71] Ye-yi. Wang. *Grammar Inference and Statistical Machine Translation.* PhD thesis, Carnegie Mellon University, 1998.

[72] Y. Wilks. Inducing adequate grammars from electronic texts. Epsrc ropa grant gr/k/66215 final report, University of Sheffield, December 1998.

[73] W. A. Woods. Transition Network Grammars for Natural Language Analysis. *Communications of the ACM*, 13(10):591–606, 1970.

[74] T. Yokomori. Polynomial-time Learning for Very Simple Grammars from Positive Data. In *Proceedings of the 4th COLT*, pages 213–227, 1991.