Formal and Empirical Grammatical Inference

Jeffrey Heinz, Colin de la Higuera and Menno van Zaanen

heinz@udel.edu, cdlh@univ-nantes.fr, mvzaanen@uvt.nl

Outline of the tutorial

- I. Formal GI and learning theory (de la Higuera)
- II. Empirical approaches to regular and subregular natural language classes (Heinz)
- III. Empirical approaches to nonregular natural language classes (van Zaanen)

I Formal GI and learning theory

- What is grammatical inference?
- What does learning or having learnt imply?
- Reasons for considering formal learning
- Some criteria to study learning in a probabilistic and a non probabilistic setting

A simple definition

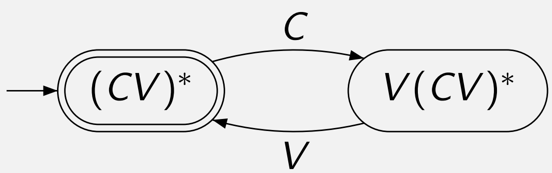
Grammatical inference is about learning a grammar given information about a language

Vocabulary

- Learning = building, inferring
- Grammar= finite representation of a possibly infinite set of strings, or trees, or graphs
- Information=you can learn from text, from an informant, by actively querying
- Language= possibly infinite set of strings, or trees, or graphs

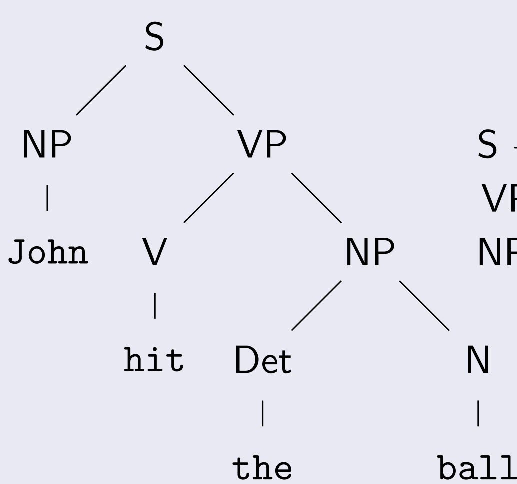
A Dfa (Ack: Jeffrey Heinz)

The (CV)* language representing licit sequences of sounds in many languages in the world. Consonants and vowels must alternate; words must begin with C and must end with V. States show the regular expression indicating its "good tails".



A context free grammar and a parse tree

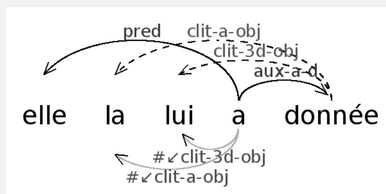
(de la Higuera 2010)



S $\rightarrow$ NP VP
VP$\rightarrow$ V NP
NP$\rightarrow$ Det N

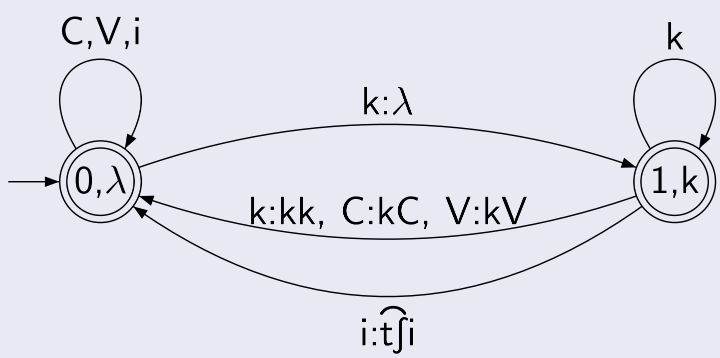
A categorial dependency grammar (Béchet et al. 2011)

- $elle \mapsto [pred]$,
- $la \mapsto [\#(\swarrow clit-a-obj)]^{\swarrow clit-a-obj}$,
- $lui \mapsto [\#(\swarrow clit-3d-obj)]^{\swarrow clit-3d-obj}$,
- $a \mapsto [\#(\swarrow clit-3d-obj)\backslash\#(\swarrow clit-a-obj)\backslash pred\backslash S/aux-a-d]$,
- $donnée \mapsto [aux-a-d]^{\nwarrow clit-3d-obj \nwarrow clit-a-obj}$



A finite state transducer (Ack: Jeffrey Heinz)

A subsequential transducer illustrating a common phonological rule of palatalization (k $\longrightarrow$ t͡ʃ / ____ i). States are labelled with a number and then the output string given by the σ function for that state.



$\Sigma = \{C, V, k, i\}$

So for example:

w	$t(w)$
kata	kata
kita	t͡ʃita
tak	tak
taki	tat͡ʃi
...	

Our definition

Grammatical inference is about learning a grammar given information about a language

Questions

- Why grammar and not language?
- Why a and not the?

Why not write "learn a language"?

Because you always learn a *representation* of a language

Paradox

Take two learners learning a context-free language, one is learning a quadratic normal form and the other a Greibach normal form, they cannot agree that they have learnt the same thing (*undecidable question*).

Worth thinking about... is it a paradox? Do two English speakers agree they speak the same language?

Our definition

Grammatical inference is about learning a grammar given information about a language

How can *a* become *the*?

- Ask for the grammar to be the smallest, best (re a score). $\rightarrow$ Combinatorial characterisation
- The learning problem becomes an optimisation problem!
- Then we often have theorems saying that
  - If our algorithm does solve the optimisation problem, what we have learnt is correct
  - If we can prove that we can't solve the optimisation problem, then the class is not learnable

Optimal with respect of some score

Score should take into account:

- Simplicity
- Coverage
- Usefulness

What scores?

- Occam argument
- Compression argument
- Kolmogorov complexity
- MDL argument

Moreover

- GI is not only about building a grammar from some data. It is concerned with saying something about:
  - the quality of the result,
  - the quality of the learning process,
  - the properties of the process.

Naive example

- Suppose you are building a random number generator.
- How are you convinced that it works?
  - Because it follows sound principles as defined by number theory specialists?
  - Because you have tested and the number 772356191 has been produced?
  - Because you have proved that the series of numbers that will be produced is incompressible?
- Empirical approach
- Experimental approach
- Formal approach

Empirical approach: using good (safe?) ideas

- For example, genetic algorithms or neural networks
- Or some mathematical *principle* (Occam, Kolmogorov, MDL,. . .)
- Can become a *principled* approach

Alternative point of view

Empirical approach is about imitating what nature (or humans) do

Experimental approach

- Benchmarks
- Competitions

- Necessary but not sufficient
- How do we know that all the cases are covered?
- How do we know that we dont have a hidden bias?

Formal approach: showing that the algorithm has converged

- Is impossible:
  - Just one run
  - Can't prove that 23 is random
- But we can say something about the algorithm:
  - That in the near future, given some string, we can predict if this string belongs to the language or not;
  - Choose between defining clearly "near future" and accepting probable truths (or error bounds) or leaving it undefined and using identification.

What else would we like to say?

That if the solution we have returned is not good, then that is because the initial data was bad (insufficient, biased)

Idea:

Blame the data, not the algorithm

Suppose we cannot say anything of the sort?

- Then that means that we may be terribly wrong even in a favourable setting
- Thus there is a *hidden bias*
- Hidden bias: the learning algorithm is supposed to be able to learn anything inside class $\mathcal{L}_1$, but can really only learn things inside class $\mathcal{L}_2$, with $\mathcal{L}_2 \subset \mathcal{L}_1$

Saying something about the process itself

- Key idea: if there is something to learn and the data is not corrupt, then, given enough time, we will learn it
- Replace the notion of *learning* by that of *identifying*

In practise, does it make sense?

- No, because we never know if we are in the ideal conditions (something to learn + good data + enough of it)
- Yes, because at least we get to blame the data, not the algorithm

Complexity issues

- Complexity theory should be used: the total or update runtime, the size of the data needed, the number of mind changes, the number and weight of errors. . .
- . . . should be measured and limited.

A linguistic criterion

- One argument appealing to linguists (we hope) is that if the criteria are not met for some class of languages that a human is supposed to know how to learn, something is wrong somewhere
- (preposterously, the maths can't be wrong. . .)

Non probabilistic settings

- Identification in the limit
- Resource bounded identification in the limit
- Active learning (query learning)

Identification in the limit

- Information is presented to the learner who updates its hypothesis after inspecting each piece of data
- At some point, always, the learner will have found the correct concept and not change from it

(Gold 1967 & 1978)

Example

Number	Presentation	Analysis of hypothesis	New hypothesis (*regexp*)
1	a +		a
2	aaa +	inconsistent	a^*
3	$aaaa$ -	inconsistent	$a(aa)^*$
4	$aaaaaa$ -	consistent	$a(aa)^*$
9234	$aaaaaaaa$ -	consistent	$a(aa)^*$
45623416	$aaaaaaaaa$ +	consistent	$a(aa)^*$

A presentation is

a function $\phi : \mathbb{N} \rightarrow X$

- where X is some set,
- and such that ϕ is associated to a language L through a function $\textsc{Yields} : \textsc{Yields}(\phi) = L$
- If $\phi(\mathbb{N}) = \psi(\mathbb{N})$ then $\textsc{Yields}(\phi) = \textsc{Yields}(\psi)$

text presentation

- A *text* presentation of a language $L \subseteq \Sigma^\star$ is a function $\phi : \mathbb{N} \to \Sigma^\star$ such that $\phi(\mathbb{N}) = L$
- ϕ is an infinite succession of all the elements of L
- (note : small technical difficulty with $\emptyset$)

informed presentation

- An *informed* presentation (or an informant) of $L \subseteq \Sigma^\star$ is a function $\phi : \mathbb{N} \to \Sigma^\star \times \{-, +\}$ such that $\phi(\mathbb{N}) = (L, +) \cup (\overline{L}, -)$
- ϕ is an infinite succession of all the elements of $\Sigma^\star$ labelled to indicate if they belong or not to L

Active presentation

- The learner interacts with the environment (modelled as an oracle) through queries
- A membership query
  - Learner presents string x
  - Oracle answer yes or no
- A correction query (Becerra-Bonache et al. 2005 & 2008)
  - Learner presents string x
  - Oracle answer yes or returns a close correction
- An equivalence query
  - Learner presents hypothesis H
  - Oracle answer yes or returns a counter-example

Example: presentations for $\{a^n b^n : n \in \mathbb{N}\}$

- Legal presentation from text: λ, a^2b^2, a^7b^7,...
- Illegal presentation from text: *ab*, *ab*, *ab*,...
- Legal presentation from informant : $(\lambda, +)$, $(abab, -)$, $(a^2b^2, +)$, $(a^7b^7, +)$, $(aab, -)$, $(abab, -)$,...

Example: presentation for Spanish

- Legal presentation from text: *En un lugar de la Mancha...*
- Illegal presentation from text: *Goooool*
- Legal presentation from informant : (*en*,+), (*whatever*,-), (*un*,+), (*lugar*,+), (*lugor*,-), (*xwszrrzt*,-),

What happens before convergence?

On two occasions I have been asked [by members of Parliament], 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.
Charles Babbage

Further definitions

- Given a presentation ϕ, ϕ_n is the set of the first n elements in ϕ.
- A learning algorithm (learner) **A** is a function that takes as input a set ϕ_n and returns a grammar of a language.
- Given a grammar G, $\mathbb{L}(G)$ is the language generated/recognised/ represented by G.

Convergence to a hypothesis

- **A** converges to G with ϕ if
  - $\forall n \in \mathbb{N} : \mathbf{A}(\phi_n)$ halts and gives an answer
  - $\exists n_0 \in \mathbb{N} : n \geq n_0 \implies \mathbf{A}(\phi_n) = G$
- If furthermore $\mathbb{L}(G) = \textsc{Yields}(\phi)$ then we have identified.

Identification in the limit

$$
\begin{tikzcd}
\mathbf{Pres}(\mathcal{L}) \arrow[rr, "\textsc{Yields}"] \arrow[dr, "\mathbf{A}"'] & & \mathcal{L} \\
& \mathcal{G} \arrow[ur, "\mathbb{L}"'] &
\end{tikzcd}
$$

Figure: The learning setting.

from (de la Higuera 2010)

Consistency and conservatism

- We say that the learner **A** is *consistent* if ϕ_n is consistent with $\mathbf{A}(\phi_n)\ \forall n$
- A consistent learner is always consistent with the past

Consistency and **conservatism**

- We say that the learner **A** is *conservative* if whenever $\phi(n+1)$ is consistent with $\mathbf{A}(\phi_n)$, we have $\mathbf{A}(\phi_n) = \mathbf{A}(\phi_{n+1})$
- A conservative learner doesn't change his mind needlessly

Learning from data

- A learner is order dependent if it learns something different depending on the order in which it receives the data.
- Usually an order independent learner is better.

What about efficiency?

- We can try to bound
  - global time
  - update time
  - errors before converging (IPE)
  - mind changes (MC)
  - queries
  - good examples needed (characteristic samples)

(Pitt 1989, de la Higuera et al. 2008)

Definition: polynomial number of implicit prediction errors

- Denote by $G \not\models x$ if G is incorrect with respect to an element x of the presentation (i.e. the learner producing G has made an implicit prediction error.

$\mathcal{G}$ is polynomially identifiable in the limit from **Pres** if there exists an identification learner **A** and a polynomial $p()$ such that given any G in $\mathcal{G}$, and given any presentation ϕ of $\mathbb{L}(G)$,
$\sharp i : \mathbf{A}(\phi_i) \not\models \phi(i+1) \leq p(|G|)$.

Definition: polynomial characteristic sample

$\mathcal{G}$ has polynomial characteristic samples for identification learner **A** if there exists a polynomial $p()$ such that: given any G in $\mathcal{G}$, $\exists Y$ correct sample for G, such that whenever $Y \subset \phi_n$, $\mathbf{A}(\phi_n) \equiv G$ and $\|Y\| \leq p(\|G\|)$

- As soon as the CS is in the data, the result is correct;
- The CS is small.

Polynomial queries

(Angluin 1987)

- Algorithm **A** learns with a polynomial number of queries if the number of queries made before halting with a correct grammar is polynomial in
  - the size of the target,
  - the size of the information received.

Main negative results

- Cannot learn NFA, CFGs from an informant in most polynomial settings (Pitt 1989, de la Higuera 1997)
- Cannot learn DFA from text (Gold 1967)
- Cannot learn DFA from membership nor equivalence queries (Angluin 1981 & 1987).

Main positive results

- Can learn DFA from an informant with polynomial resources (Oncina and García 1992);
- Can learn DFA from membership and equivalence queries (Angluin 1987).

Probabilistic settings

- PAC learning (about learning yes-no machines with fixed but unknown distributions)
- Identification with probability 1 (about identifying distributions)
- PAC learning distributions (about approximately learning distributions)

Learning a language from sampling

- We have a distribution over $\Sigma^\star$
- We sample twice:
  - once to learn,
  - once to see how well we have learned
- The PAC setting: Les Valiant, Turing award 2010

PAC-learning

(Valiant 1984, Pitt 1989)

- $\mathcal{L}$ a class of languages
- $\mathcal{G}$ a class of grammars
- $\epsilon > 0$ and $\delta > 0$
- m a maximal length over the strings
- n a maximal size of machines

H is ϵ-AC (approximately correct)*

if

$$Pr_D[H(x) \neq G(x)] < \epsilon$$

Polynomial PAC learning

There is a polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that

- in order to learn ϵ-AC machines of size at most n with error at most δ we require at most $p(m, n, \frac{1}{\delta}, \frac{1}{\delta})$ data and time;
- we want the errors to be less than ϵ and bad luck to be less than δ.

(French radio)

- Unless there is a surprise there should be no surprise
- French radio, (after the last primary elections, on 3rd of June 2008)
- First surprise is δ, second surprise is ϵ

Formal GI and learning theory GI of Regular Patterns Empirical GI and nonregular patterns

Results (Kearns and Valiant 1989, Kearns and Vazirani 1994)

- Using cryptographic assumptions, we cannot PAC-learn DFA
- Cannot PAC-learn NFA, CFGs with membership queries either
- Learning can be seen as finding the encryption function from examples (Kearns & Vazirani)

Alternatively

- Instead of learning classifiers in a probabilistic world, learn directly the distributions!
- Learn probabilistic finite automata (deterministic or not)

No error (Angluin 1988)

- This calls for identification in the limit with probability 1
- Means that the probability of not converging is 0
- Goal is to identify the structure and the probabilities
- Mainly a (nice) theoretic setting

Results

- If probabilities are computable, we can learn with probability 1 finite state automata (Carrasco and Oncina, 1994)
- But not with bounded (polynomial) resources (de la Higuera and Oncina, 2004)

With error

- Pac definition applies
- But error should be measured by a distance between the *target distribution* and the *hypothesis*
- How do we measure the distance: L_1, L_2, L_∞, Kullback-Leibler?

Results

- Too easy to learn with L_∞
- Too hard to learn with L_1
- Both results hold for the same algorithm! (de la Higuera and Oncina, 2004)
- Nice algorithms for biased classes of distributions

Open problems

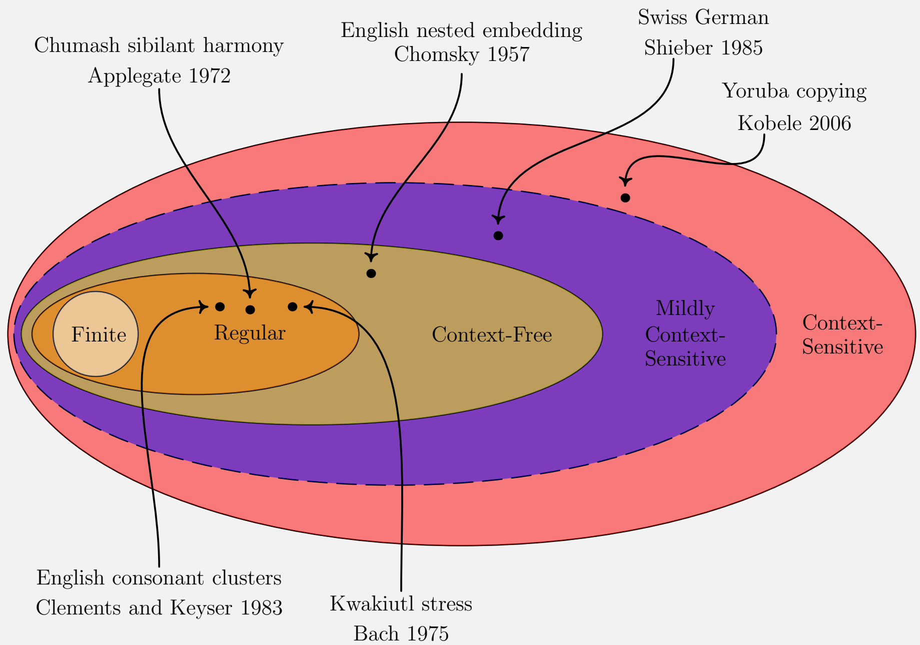
We conclude this section on "what is language learning about" with some open questions:

- What is a good definition of polynomial identification?
- How do we deal with shifting targets? (robustness issues)
- Alternative views on learnability?
- Is being learnable a good indicator of being linguistically reasonable?
- Can we learn transducers? Probabilistic transducers?

II. GI of Regular Patterns

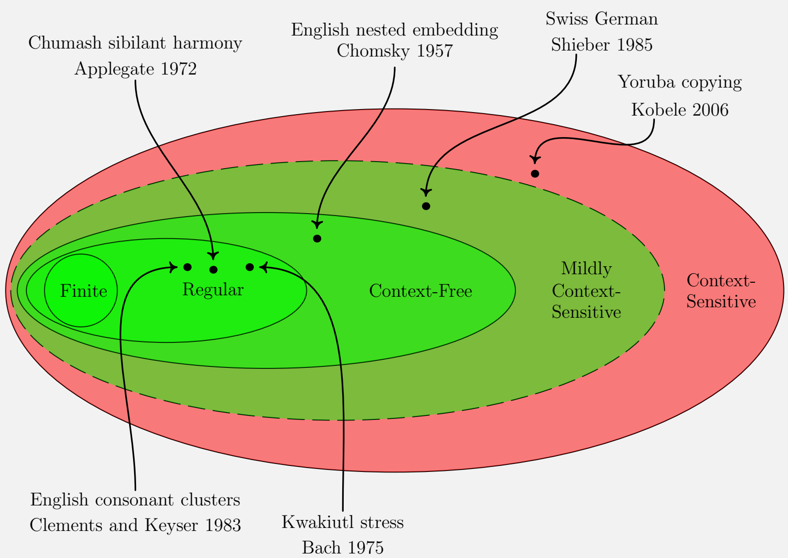
- Why regular?
- What are the general GI strategies?
- What are the main results?
- The main techniques?
- The main lessons?

Logically Possible Computable Patterns



GI Strategies

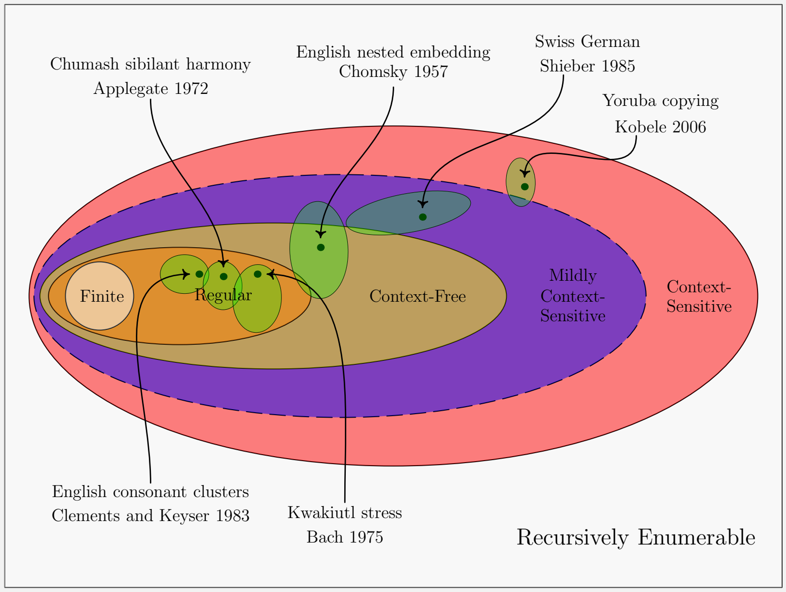
#1. Define “learning” so that large regions can be learned



GI Strategies

#2. Target non-superfinite cross-cutting classes

(instructor's bias)



Common Theme

1. Different learning frameworks may *better characterize* the data presentations learners actually get (strategy #1).
2. Classes of formal languages may exist which *better characterize* the patterns we are interested in (strategy #2).
3. Hard problems are *easier to solve* with better characterizations because the instance space of the problem is smaller.

Why Begin with Regular?

Insights obtained here can be (and have been) applied fruitfully to nonregular classes.

- Angluin 1982 showed a subclass of regular languages (the reversible languages) was identifiable in the limit from positive data by an incremental learner.
- Yokomori's (2004) Very Simple Languages are a subclass of the context-free languages, but draws on ideas from the reversible languages.
- Similarly, Clark and Eryaud's (2007) substitutable languages (also subclass of context-free) are also based on insights from this paper.

Section Outline

1. Targets of Learning
2. Learning Frameworks
3. State-merging
4. Results for learning regular languages, relations, and distributions

Targets of Learning: Regular Languages

Multiple grammars (i.e. representations) for regular languages:

1. Regular expressions
2. Generalized regular expressions
3. Finite state acceptors
4. Words which satisfy formulae in monadic second order logic
5. Right or left branching rewrite rules
6. ...

Targets of Learning: Regular Relations

Multiple grammars (i.e. representations) for regular relations:

- Regular expressions (for relations)
- Generalized regular expressions (for relations)
- Finite state transducers
- ...

Targets of Learning: Regular distributions

Multiple grammars (i.e. representations) for distributions over regular sets and relations:

- Weighted finite state automata
- Hidden Markov Models
- Weighted right or left branching rewrite rules
- ...

This tutorial: Finite State Automata

Acceptors and subsequential transducers admit canonical forms

1. The smallest deterministic acceptor, syntactic monoids, ...
2. Canonical forms relate to algebraic properties (Nerode equivalence relation, i.e. states represent sets of "good tails")
3. In contrast, canonical regular expressions have yet to be determined. For example, there are no canonical (e.g. shortest) regular expressions for regular languages.

Learning Frameworks: Main Choices

Success required on which input data streams?

All possible vs. some restricted set

i.e. "distribution-free" vs. "non distribution-free"

What kind of samples?

Positive data vs. postive and negative data

- Other choices (e.g. query learning) are not discussed here.

Learning Frameworks: Main Results

"Distribution-free" w/ positive and negative data

1. The class of r.e. languages is identifiable in the limit (Gold 1967)
2. Non-enumerative algorithms for regular languages:
   1. Gold (1978)
   2. RPNI (Oncina and García 1992)

Learning Frameworks: Main Results

"Distribution-free" with positive data only

1. No superfinite class (including regular, cf, etc.) is identifiable in the limit (Gold 1967)
2. Not even the finite class is PAC-learnable (Blumer et al. 1989)
3. No superfinite class is identifiable in the limit with probability p ($p > 2/3$) (Pitt 1985, Wiehagen et al. 1986, Angluin 1988)
4. But many subregular classes are learnable in this difficult setting.

Formal GI and learning theory GI of Regular Patterns Empirical GI and nonregular patterns

Learning Frameworks: Main Results

"Distribution-free" with positive data only: learnable subregular classes

1. reversible languages (Angluin 1982)
2. strictly local languages (Garcia et al. 1990)
3. locally testable and piecewise testable (Garcia and Ruiz 2004)
4. left-to-right and right-to-left iterative languages (Heinz 2008)
5. strictly piecewise languages (Heinz 2010)
6. ...
7. subsequential functions (Oncina et al. 1993)
8. ...

Learning Frameworks: Main Results

“Non distribution-free” w/ positive data only

1. The class of r.e. languages are identifiable in the limit from computable classes of r.e. texts (Gold 1967)
2. The class of r.e. distributions are identifiable from “approximately computable” sequences (Angluin 1988, Chater and Vitanyí 2007)
3. The class of distributions describable with Probabilistic Deterministic FSAs (PDFAs) is learnable with probability one (de la Higuera and Thollard 2000)
4. The class of distributions describable with PDFAs is learnable in a modified PAC setting (Clark and Thollard, 2004)

Learning regular languages: Key technique

State-merging

- Angluin 1982 (reversible languages)
- Muggleton 1990 (contextual languages)
- Garcia et al. 1990 (strictly local languages)
- Oncina et al. 1993 (subsequential functions)
- Clark and Thollard 2004 (PDFA distributions)
- ...

Other techniques

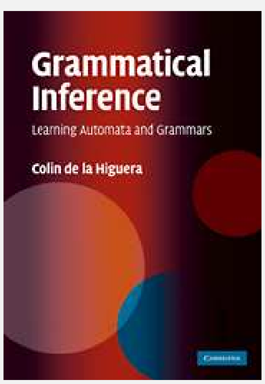
Lattice-climbing

- Heinz 2010 (strictly local languages, strictly piecewise languages, many others)
- Kasprizk and Kötzing 2010 (function-distinguishable lanaguages, pattern languages, many others)

State-splitting

- Tellier (2008)

Only so much can be covered. . .



It's impossible to be fair to all those who have contributed and to cover all the variants, even all the algorithms in a short tutorial. That's why there are books!

Overview of State-merging

1. Builds a FSA representation of the input
2. Generalize by merging states

Illustrative Example: Stress pattern of Pintupi

a.	páɳa	‘earth’	$\boldsymbol{\acute{\sigma}}\ \sigma$
b.	tʲúʈaya	‘many’	$\boldsymbol{\acute{\sigma}}\ \sigma\ \sigma$
c.	máɭawàna	‘through from behind’	$\boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma$
d.	púɭiŋkàlatʲu	‘we (sat) on the hill’	$\boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \sigma$
e.	tʲámulìmpatʲùŋku	‘our relation’	$\boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma$
f.	ʈíɭirìŋulàmpatʲu	‘the fire for our benefit flared up’	$\boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \sigma$
g.	kúranʲùlulìmpatʲùɻa	‘the first one who is our relation’	$\boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma$
h.	yúmaɻìŋkamàratʲùɻaka	‘because of mother-in-law’	$\boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \sigma$

Generalization (Hayes (1995:62) citing Hansen and Hansen (1969:163)):

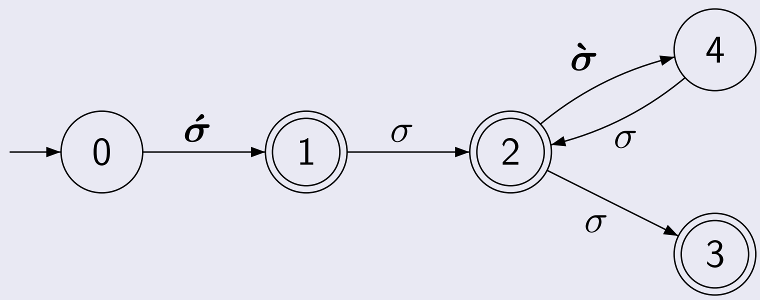
- Primary stress falls on the initial syllable
- Secondary stress falls on alternating nonfinal syllables

Illustrative Example: Stress pattern of Pintupi

Generalization (Hayes (1995:62) citing Hansen and Hansen (1969:163)):

- Primary stress falls on the initial syllable
- Secondary stress falls on alternating nonfinal syllables

Minimal deterministic FSA for Pintupi Stress



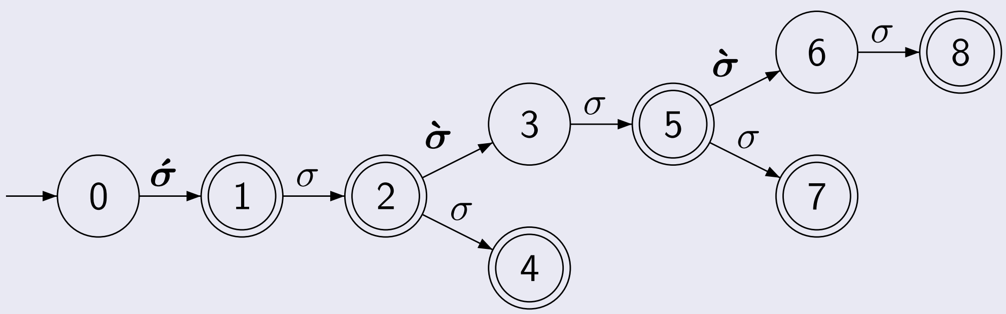
Structured representations of Input

1. Each word its own FSA (Nondeterministic)
2. Prefix Trees (deterministic)
3. Suffix Trees (reverse determinstic)

Examples of Prefix and Suffix Trees

$$S = \left\{ \begin{array}{ll} \boldsymbol{\acute{\sigma}} & \boldsymbol{\acute{\sigma}}\ \sigma \\ \boldsymbol{\acute{\sigma}}\ \sigma\ \sigma & \boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma \\ \boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \sigma & \boldsymbol{\acute{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma\ \boldsymbol{\grave{\sigma}}\ \sigma \end{array} \right\}$$

PT(S)

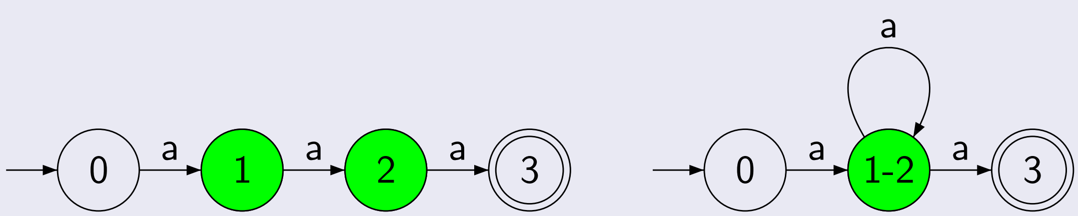


ST(S)

State-merging Informally

Eliminate redundant environments by state-merging.

- States are identified as equivalent and then *merged*.
- All transitions are preserved.
- This is one way in which generalizations may occur—because the post-merged machine accepts everything the pre-merged machine accepts, possibly more.



Machine A Machine B

- The merged machine may not be deterministic.

State-merging Formally

Definition

Given an acceptor $A = (Q, I, F, \delta)$ and a partition π of its states *state-merging* returns the acceptor $A/\pi = (Q', I', F', \delta')$:

1. $Q' = \pi$ (the states are the *blocks* of π)
2. $I' = \{B \in \pi : I \cap B \neq \emptyset\}$
3. $F' = \{B \in \pi : F \cap B \neq \emptyset\}$
4. For all $B \in \pi$ and $a \in \Sigma$,
   $\delta'(B, a) = \{B' \in \pi : \exists q \in B,\ q' \in B' \text{ such that } q' \in \delta(q, a)\}$

Theorem

Theorem

Given any regular language L, let $A(L)$ denote the minimal deterministic acceptor recognizing L. There exists a finite sample $S \subseteq L$ and a partition π over $PT(S)$ such that $PT(S)/\pi = A(L)$.

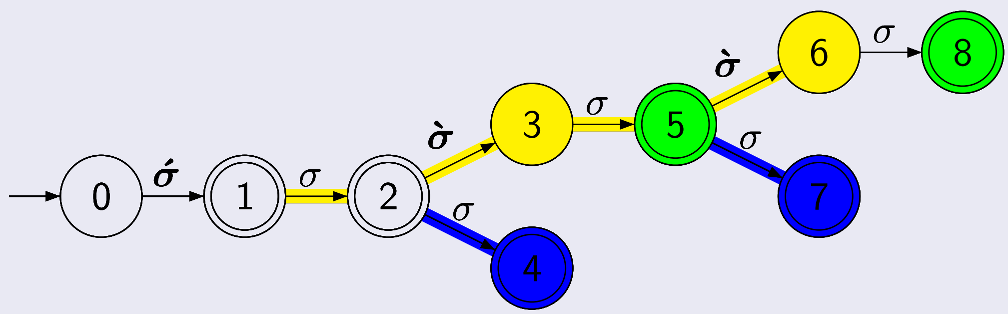
Notes

- The finite sample need only exercise every transition in $A(L)$.
- What is π?

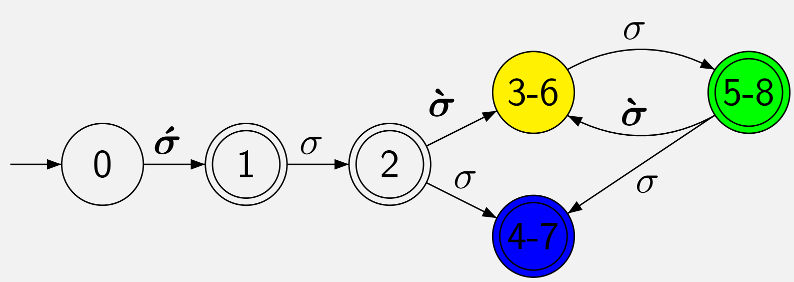
Illustrative Example

Let's merge states with the same incoming paths of length 2!

PT(S)



Result of State Merging



This acceptor is not the canonical acceptor we saw earlier but it recognizes the same language.

Generalization (Hayes (1995:62) citing Hansen and Hansen (1969:163)):

- Primary stress falls on the initial syllable
- Secondary stress falls on alternating nonfinal syllables

Summary of Algorithm

1. States in the prefix tree are merged if they have the same k-length suffix.

$$u \sim v \overset{\text{def}}{\Longleftrightarrow} \exists x, y, w \text{ such that } |w| = k, u = xw, v = yw$$

2. The algorithm then is simply:

$$G = PT(S)/\pi_{\sim}$$

3. This algorithm provably identifies in the limit from positive data the Strictly $(k+1)$-Local class of languages (Garcia et al. 1990).

Back to the Illustrative Example

Results for stress patterns more generally

- Out of 109 distinct stress patterns in the world's languages (encoded as FSAs), this state-merging strategy works for only 44 of them
- If we merge states with the same paths up to length 5(!), only 81 are learned.
- This is the case even permitting very generous input samples.

In other words, 44 attested stress patterns are Strictly 3-Local and 81 are Strictly 6-Local. 28 are not Strictly 6-Local In fact those 28 are not Strictly k-Local for any k (Edlefsen et al. 2008).

Other ways to merge states

If the current structure is "ill-formed" then merge states to eliminate source of ill-formedness

State equivalence relations

1. merge state with same incoming paths of length k (Garcia et. al 1990)
2. recursively eliminate reverse non-determinism (Angluin 1982)
3. merge states with same "contexts" (Muggleton 1990, Clark and Eryaud 2007)
4. merge final states (Heinz 2008)
5. merge states with same "neighborhood" (Heinz 2009)
6. ...
7. merge states to maximize posterior probability (for HMMs, Stolcke 1994)
8. ...

Other ways to merge states

Merge states indiscriminately unless "ill-formedness" arises

Merge unless something tells us not to

1. unless "onward subsequentiality" is lost (for transducers, Oncina et al. 1993)
2. unless they are "μ-distinguishable" (Clark and Thollard 2004)
3. ...

State-merging as inference rules

Strictly k-Local languages (Garcia et al. 1990)

merge states with same incoming paths of length k

$$\forall u, v, w \in \Sigma^* \quad : \quad uv, wv, \in \mathit{Prefix}(L) \text{ and } |v| = k$$

$$\Downarrow$$

$$\mathit{Tails}_L(uv) = \mathit{Tails}_L(wv) \in L$$

Formal GI and learning theory GI of Regular Patterns Empirical GI and nonregular patterns

State-merging as inference rules

0-Reversible languages (Angluin 1982)

recursively eliminate reverse non-determinism

$$\forall u, v, w, y \in \Sigma^* \quad : \quad uv, wv, uy \in L \Rightarrow wy \in L$$

State-merging summary

1. Distinctions maintained in the prefix tree are lost by state merging, which results in generalizations.
2. The choice of partition corresponds to the generalization strategy (i.e. which distinctions will be maintained and which will be lost)

Gleitman (1990:12):

> *The trouble is that an observer who notices everything can learn nothing for there is no end of categories known and constructible to describe a situation [emphasis in original].*

Results for regular languages

Distribution-free with positive data

Identification in the limit from positive data

1. strictly k-local languages (each state corresponds to suffixes of up to length k) (Garcia et al. 1990)
2. reversible languages (acceptors are both forward and reverse k-deterministic for some k) (Angluin 1982)
3. k-contextual languages (Muggleton 1990)
4. ...

Regular relations

Regular relations in CL

1. transliteration
2. translation
3. . . .
4. anything with finite state transducers

OSTIA (Oncina et al. 1993)

distribution-free with positive data

OSTIA

1. identifies *subsequential functions* in the limit from positive data.
2. Merges states greedily unless subsequentiality is violated
3. If the function is partial, exactness is guaranteed only where the function is defined.

OSTIA (Oncina et al. 1993)

Subsequential relations

1. are a subclass of the regular relations, recognizing functions.
2. are those which are recognized by subsequential transducers, which are determinstic on the input and which have an "output" string associated with every state.
3. have a canonical form.
4. have been generalized to permit up to *p* outputs for each input (Mohri 1997).

OSTIA for learning phonological rules

Gildea and Jurafsky 1996

1. Show that OSTIA doesn't learn the English tapping rule or German word-final devoicing rule from data present in adapted dictionaries of English or German
2. Applied additional phonologically motivated heuristics to improve state-merging choices.

What about well-defined subclasses of subsequential relations?

Weighted finite-state automata

non-distribution-free with positive data

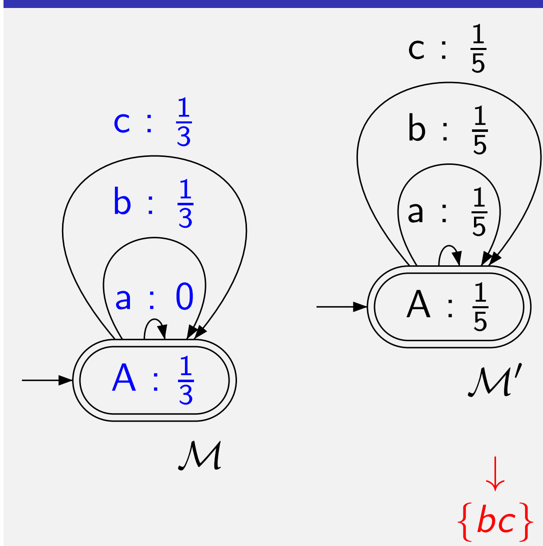
The problem

Given a finite multiset of words drawn independently from the target distribution, what grammar accurately describes the distribution?

Theorem

The class of distributions describable with Non-deterministic Probabilistic Finite-State Automata (NPFA) exactly matches the class of distributions describable with Hidden Markov Models (Vidal et al. 2005).

Maximum Likelihood Estimation



$\mathcal{M}$ represents a family of distributions with 4 parameters. $\mathcal{M}'$ represents a particular distribution in this family.

Theorem

For a sample S and deterministic finite-state acceptor $\mathcal{M}$, **counting the parse of S through $\mathcal{M}$ and normalizing at each state** *optimizes the maximum-likelihood estimate.*

(Vidal et. al 2005, de la Higuera 2010)

Strictly 2-Local Distributions are bigram models

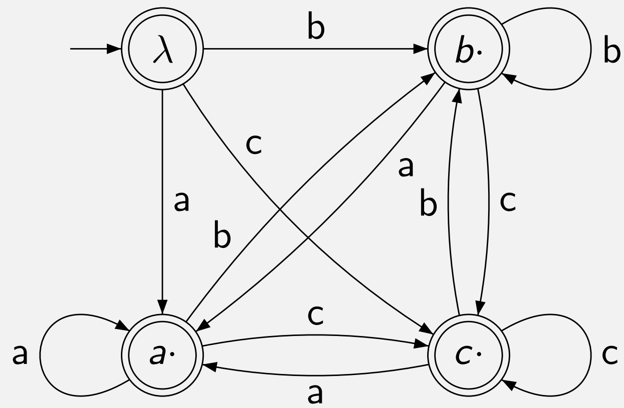
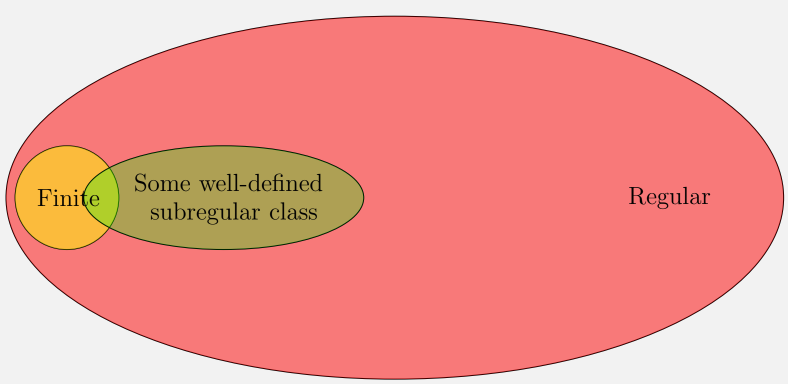


Figure: The structure of a bigram model. The 16 parameters of this model are given by associating probabilities to each transition and to "ending" at each state.

Subregular distributions



1. When the structure of a Deterministic FSA is known in advance, MLE is easy to do.
2. The DFA represents a subregular class of distributions.

Strictly Piecewise Distributions

1. *N*-gram models can't describe long-distance dependencies.

Long-distance dependencies in phonology

1. **Consonantal harmony**

   (Jensen 1974, Odden 1994, Hansson 2001, Rose and Walker 2004, and many others)
2. **Vowel harmony**

   (Ringen 1988, Baković 2000, and many others)

Sibilant Harmony example from Samala (Ineseño Chumash)

[ʃtojonowonowaʃ] 'it stood upright' (Applegate 1972:72)

cf. *[**s**tojonowonowa**ʃ**] and

cf. *[**ʃ**tojonowonowa**s**]

Hypothesis: *[stojonowonowaʃ] and *[ʃtojonowonowas] are ill-formed because the *discontiguous subsequences* *sʃ* and *ʃs* are ill-formed.

Strictly Piecewise languages

Rogers et al. 2010

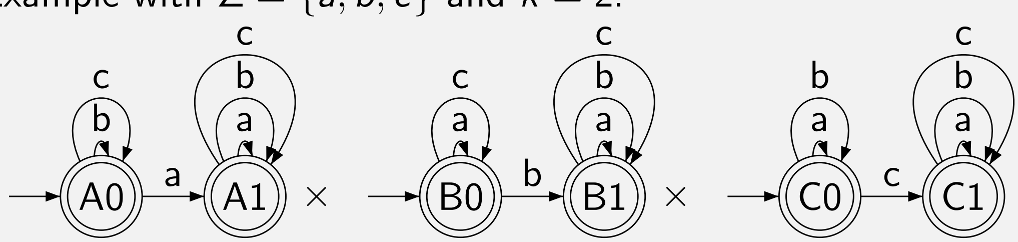
1. solely make distinctions on the basis of potentially discontiguous subsequences up to some length k
2. are mathematically natural. They have several chacterizations in terms of formal language theory, automata theory, logic, model theory, and the
3. algebraic theory of automata (Fu et al. 2011)

Strictly Piecewise Distributions

Heinz and Rogers 2010

1. are defined in terms of the factored automata-theoretic representations (Rogers et al. 2010)
2. along with the co-emission probability as the product (Vidal et al. 2005)
3. Estimation over the factors permits learnability of the patterns like the ones in Samala.

Example with $\Sigma = \{a, b, c\}$ and $k = 2$.



SP2 learning results for Chumash

- Training corpus 4800 words from a dictionary of Samala

$P(x \mid y <)$		x: s	x: t͡s	x: ʃ	x: t͡ʃ
y	s	0.0325	0.0051	0.0013	0.0002
y	t͡s	0.0212	0.0114	0.0008	0.
y	ʃ	0.0011	0.	0.067	0.0359
y	t͡ʃ	0.0006	0.	0.0458	0.0314

Table: SP2 probabilities of sibilant occuring sometime after another one (collapsing laryngeal distinctions)

Learning larger classes of regular distributions

More non-distribution-free with positive data

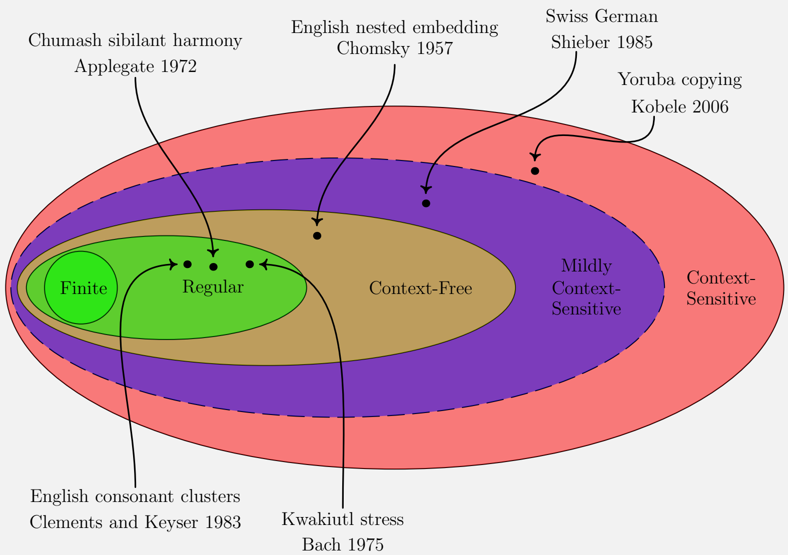
The class of distributions describable with PDFA

1. are identifiable in the limit with probability one (de la Higuera and Thollard 2000).
2. are learnable in modified-PAC setting (Clark and Thollard 2004).
3. The algorithms presented employ state-merging methods.

1. This is a (much!) larger class than that which is describable with n-gram distributions or with SP distributions.
2. To my knowledge these approaches have not been applied to tasks in CL.

Summary

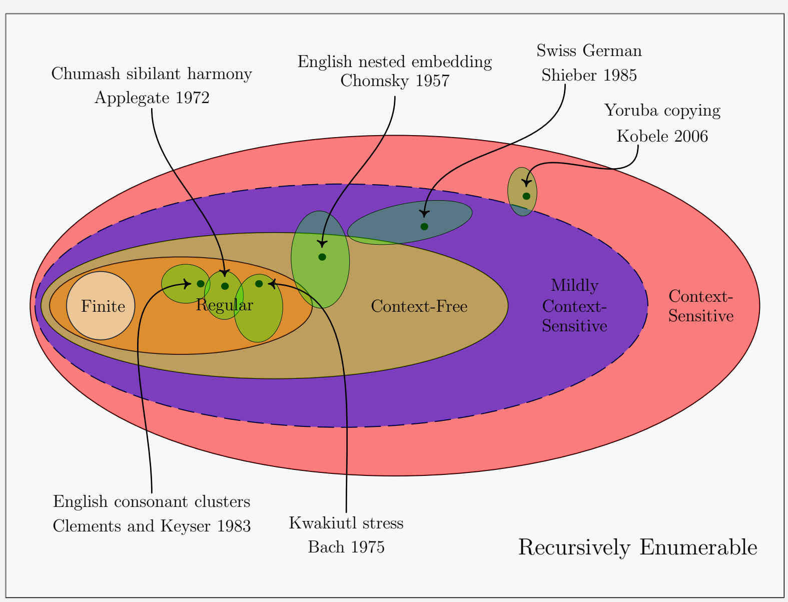
#1. Define "learning" so that large regions can be learned



Oncina et al. 1993, de la Higuera and Thollard 2000, Clark and Thollard 2004, . . .

Summary

#2. Target non-superfinite cross-cutting classes



Angluin 1982, Muggleton 1990, Garcia et al. 1990, Heinz 2010, . . .

Have we put the cart before the horse?

1. So far we have discussed algorithms that learn various classes of languages.
2. But shouldn't we *first* know *which* classes are relevant for our goals?
3. E.g. for phonology, while "being regular" may be a *necessary* property of phonological patterns, it certainly is not *sufficient*.

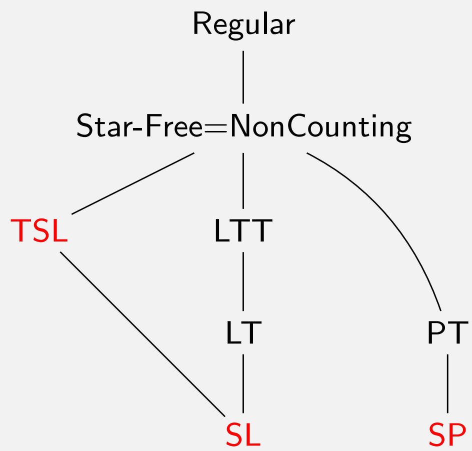
Have we put the cart before the horse?

Research strategy

Patterns $\Rightarrow$ Characterizations $\Rightarrow$ Learning algorithms

1. Identify the range and kind of patterns (linguistics).
2. Characterize the range and kind of patterns (computational linguistics).
3. Create learning algorithms for these classes, prove their success in a variety of settings, and otherwise demonstrate their success (grammatical inference, formal learning theory, computational linguistics)

Subregular classes of regular sets



Proper inclusion relationships among subregular language classes.
instructor's hunch for phonology

TSL	Tier-based Strictly Local	PT	Piecewise Testable
LTT	Locally Threshold Testable	SL	Strictly Local
LT	Locally Testable	SP	Strictly Piecewise

(McNaughton and Papert 1971, Simon 1975, Rogers and Pullum 2007, in press, Rogers et al. 2010, Heinz et al. 2011)

Conclusion to section 2 part 1

1. State-merging is a well-studied strategy for inferring automata, including acceptors, transducers, and weighted acceptors and transducers.
2. It has yielded theoretical results in many learning frameworks including both distribution-free and non-distribution-free learning frameworks.

Conclusion to section 2 part 2

1. Many subclasses of regular languages are learnable even in the hardest learning settings.
2. Recent advances yield algorithms for large classes (probabilistic DFAs)
3. Computational linguists can explore which are relevant to natural language and consequently which are useful for NLP!
4. There is a rich literature in GI which speaks to these classes, and how such patterns in these classes can be learned.

Overview

- Empirical grammatical inference
- Family of languages
- Information contained in input
- Overview of systems
- Evaluation issues
- From empirical to formal GI

Introduction

- Language learning
  - Starting from family of languages
  - Given set of samples
  - Identify language that is used to generate samples
- Formal grammatical inference
  - Identify family of languages that can be learned efficiently
  - Under certain restrictions
- Empirical grammatical inference
  - Exact underlying family of languages is unknown
  - Target language is approximation

Empirical GI

- Try to identify language given samples
  - E.g. sentences (syntax), words (morphology), ...
- Underlying language class is unknown
  - For algorithm we still need to make a choice
- If identification is impossible, provide approximation
  - Evaluation of empirical GI is different from formal GI

Family of languages

- What is the underlying family of languages?
- Choice has impact on learning algorithm
- Many possibilities
  - Use simple, fixed structures (n-grams)
    - Find probabilities
  - Extract structure from treebanks
    - Slightly more flexible structure
    - Find probabilities
  - Learn structure
    - Flexible structure
    - Find probabilities

N-grams

1. Starting from a plain text or collection of texts (corpus)
2. Extract all subsequences of length n (n-grams)
3. Count occurrences of n-grams in texts
4. Assign probabilities to each n-gram based on counts

Issues

- Unseen n-grams
  - Back-off: use n-grams with smaller n
  - Smoothing: adjust probabilities for unseen n-grams

Using n-gram models

- How likely is the sentence 'John likes Mary'?
  - Unigram language model
    - $P(\text{John likes Mary}) \approx P(\text{John})P(\text{likes})P(\text{Mary})$
  - Bigram language model
    - $P(\text{John likes Mary}) \approx P(\text{John}|\langle s\rangle)P(\text{likes}|\text{John})P(\text{Mary}|\text{likes})$
  - Trigram language model
    - $P(\text{John likes Mary}) \approx$ $P(\text{John}|\langle s\rangle\langle s\rangle)P(\text{likes}|\langle s\rangle\text{John})P(\text{Mary}|\text{John likes})$
  - N-gram language model
    - $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1}^{k-1})$
- N-grams provide a probability for each sequence
  - Probability describes how well sequence fits language

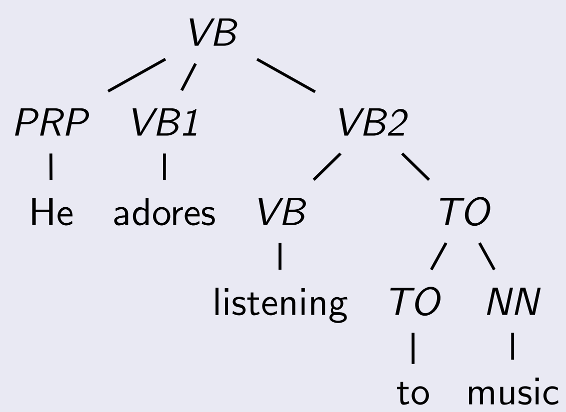
Extract structure from treebanks

1. Starting from a treebank (sentences with structure)
2. Extract grammar rules that are used to create tree structures
   - For instance, context-free grammars (Charniak 1993)
   - or sub-trees (Data-Oriented Parsing) (Bod 1998)
3. Count occurrences of grammar rules in treebank
4. Assign probabilities to grammar rules based on counts

Issues

- Over-generalization, "incorrect" probabilities
  - Add information on applicability of grammar rules (Johnson 1998)
  - Reestimate probabilities (EM) (Dempster et al 1977, Lari and Young 1990)

Extract structure from tree



VB →*PRP VB1 VB2*
PRP→He
VB1→adores
VB2→*VB TO*
VB →listening
TO →*TO NN*
TO →to
NN →music

- Extract counts from treebank → probabilities
- Reestimate probabilities
  - Improve fit of grammar and sentences

Learn structure

1. Starting from a corpus
2. Identify regularities that may serve as grammar rules
3. Output:
   - Structure assigned to sentences → extract grammar
   - Extracted grammar rules (and probabilities) → parse

Issues

- Learning system has to deal with both
  - flexibility in structure
  - probabilities of structure

Summarizing fixed versus flexible structure

- Fixed versus flexible is really a sliding scale
- Language modelling using *n*-grams
  - Structure is very simple and very rigid
  - Requires plain sequences as input
  - Corresponds to *k*-testable languages (García 1990)
- Language modelling using extracted grammar rules
  - Structure is more flexible, but restricted by treebank
  - Requires structured sequences as input
  - Corresponds to e.g. (limited) context-free languages
- *"Learning structure"*
  - Structure is flexible, restricted by learning algorithm
  - Requires plain sequences as input
  - Corresponds to e.g. context-free languages

Empirical grammatical inference

- Choices:
  - What type of grammar are we learning?
    - Regular language
    - *K*-testable language (*n*-grams)
    - Context-free language
    - ...
  - What kind of input do we require?
    - Sequence of words (sentence)
    - Sequence of part-of-speech tags
    - (Partial) tree structures
    - ...
  - What kind of output do we want?
    - Structured version of input
    - Explicit grammar
    - Binary or *n*-ary (context-free rules)
    - ...

Overview of systems

- EMILE
- Alignment-Based Learning (ABL)
- ADIOS
- CCM+DMV
- U-DOP
- ...

Underlying approach

- Given a collection of plain sentences
- On what basis are we going to assign structure?
- Should structure be linguistically motivated?
  - or similar to what linguists would assign?
- Perhaps we can use tests for constituency to find structure

Substitutability

Elements of the same type are substitutable

Test for constituency (Harris, 1951)

What is (a family fare)$_{NP}$

Replace noun phrase with another noun phrase

What is (the payload of an African Swallow)$_{NP}$

Learning by reversing test

What is (*a family fare*)$_X$

What is (*the payload of an African Swallow*)$_X$

EMILE

- Learns context-free grammars
- Using plain sentences
- Originally used to show formal learnability of (a form of) Categorial Grammars in a PAC learning setting
- (Adriaans 1992, Adriaans and Vervoort 2002, Vervoort 2000)

Approach

1. Starting from simple sentences
   identify recurring subsequences
2. Store recurring subsequences and contexts
3. Introduce grammar rules when there is enough evidence
   - Practical implementation allows for several constraints
     - Context length, subsequence length, ...

Example matrix

John walks
Mary walks
John sees Mary

	(.) walks	John (.)	(.) sees Mary	...	*contexts*
John	x		x	...	
walks		x		...	
Mary	x			...	
sees				...	
⋮	⋮	⋮	⋮	⋱	
terms					

Learn grammar rules

- Terms that share (approximately) same context are clustered
  - "John" and "Mary" are grouped together
- Occurrences of terms in cluster are replaced by new symbol
- Modified sequences may again contain terms/contexts
- Terms may consist of multiple words

Example

John walks $\Rightarrow X$ walks
Mary walks $\Rightarrow X$ walks
John sees Mary $\Rightarrow X$ sees X

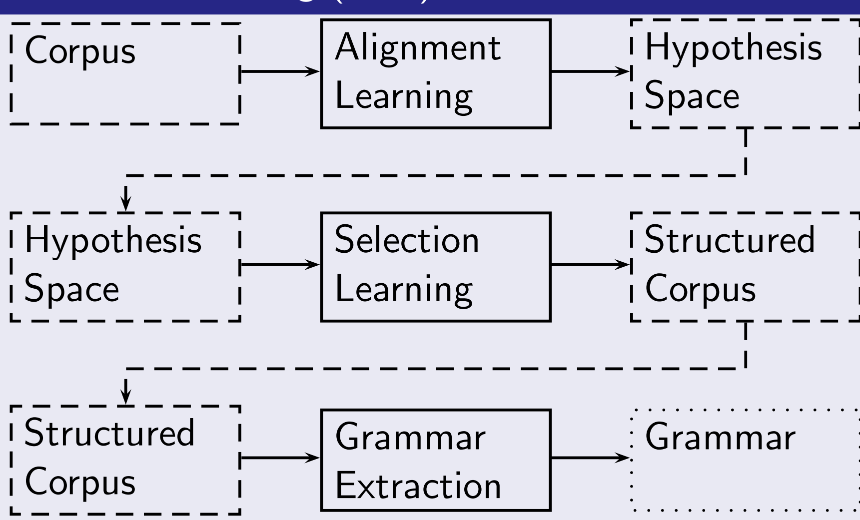
Mary slaps John $\Rightarrow X$ slaps X

"sees" and "slaps" now also share the same context

Alignment-Based Learning (ABL)

- Based on substitutability test
- Using plain sentences
- Similar to EMILE, but
  - Clustered terms are not explicitly replaced by symbol
  - Terms and contexts are *always* separated
  - All terms are considered (and only selected afterwards)
- Output is structured version of input or grammar
- (van Zaanen 2000a, b, 2002)

Alignment-Based Learning (ABL)



Alignment-Based Learning (ABL)

- Alignment learning
  - Align pairs of sentences
  - Unequal parts of sentences are stored as hypotheses
- (Clustering)
  - Group hypotheses in same context together
- Selection learning
  - Remove overlapping hypotheses

Alignment learning

- Align pairs of sentences
  - using edit distance (Wagner and Fischer 1974)
  - or suffixtrees (Geertzen and van Zaanen 2004, Ukkonen 1995)
- Unequal parts of sentences are stored as hypotheses
- Align all sentences in a corpus to all others

Example

$(_{Y_1}$ I need $(_{X_1}$ a dinner during the flight$)_{X_1})_{Y_1}$
$(_{Z_1}$ I need$)_{Z_1}$ $(_{X_1}\underline{\text{to return on}}$ $(_{Z_2}$tuesday$)_{Z_2})_{X_1}$
$(_{Y_1}(_{Z_1}$he wants$)_{Z_1}$ $\underline{\text{to return on}}$ $(_{Z_2}$wednesday$)_{Z_2})_{Y_1}$

Selection Learning

- Alignment learning can generate overlapping brackets
- Underlying grammar is considered context-free
- Structure describes parse according to underlying grammar
- "Wrong" brackets have to be removed
  - Based on e.g. chronological order or statistics

Example

from $(_{Y_1}$Tilburg $(_{X_2}$to$)_{Y_1}$ Portland$)_{X_2}$
from $(_{X_1}$Portland $(_{Y_2}$to$)_{X_1}$ Tilburg$)_{Y_2}$

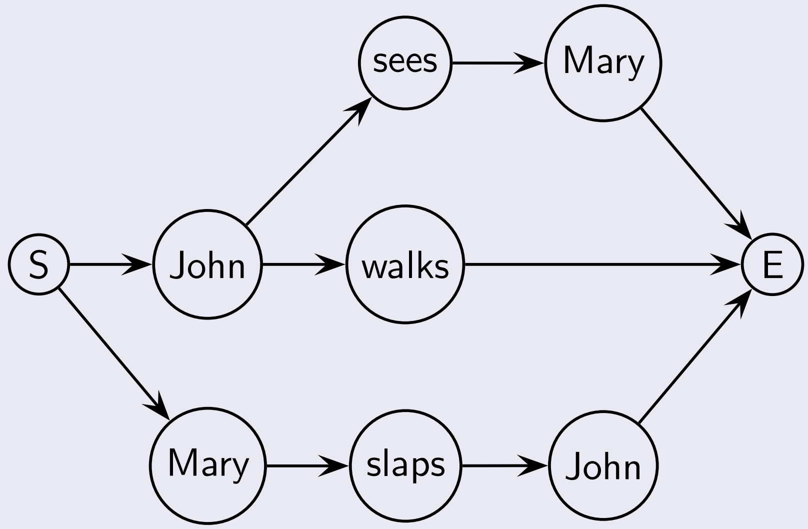
ADIOS

- Automatic Distillation of Structure (ADIOS) (Solan 2005)

Idea

1. Represent language as a graph
2. Compress graph
3. As long as possible, find significant patterns in paths
   - Using substitutability and *significance tests*
4. (Recursion may be added as a post-processing step)

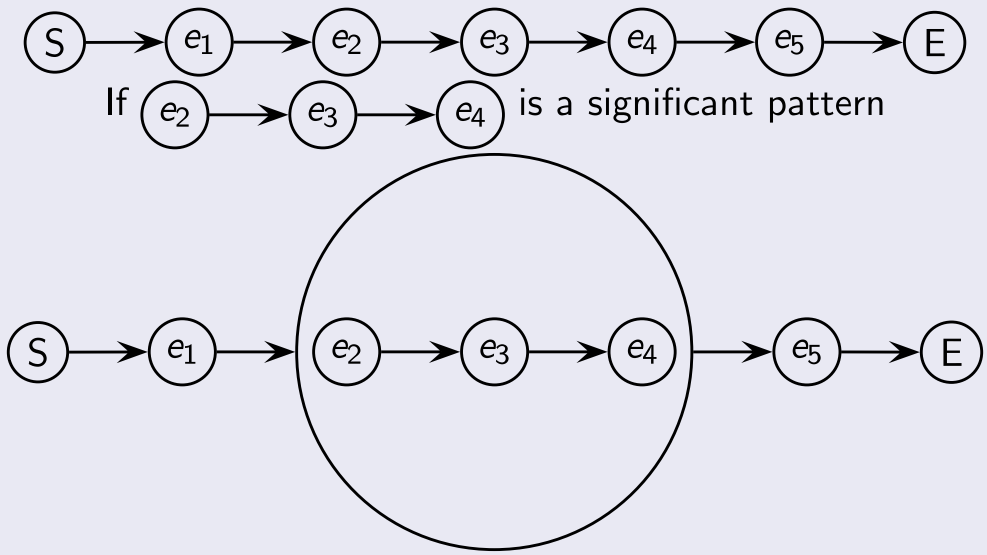
Graph



Phases

1. Initialization
   - Load all sentences (as paths) in the graph
2. Pattern distilation
   - Find sub-paths
     - shared by significant number of partially-aligned paths
     - using motif-extraction (MEX) algorithm
3. Generalization
   - Group all nodes that occur in same pattern together
   - Cluster words/subsequences similarly to EMILE
4. Repeat 2 and 3 until no new patterns are found

Graph



MEX

- Compute probabilities depending on in-/out-degree of nodes

$$P_R(e_1; e_2) = \frac{\#\text{ paths from } e_1 \text{ to } e_2}{\#\text{ paths to } e_1}$$

$$P_R(e_1; e_3) = \frac{\#\text{ paths from } e_1 \text{ to } e_3}{\#\text{ paths to } e_1}$$

$$D_R(e_1; e_3) = \frac{P_R(e_1; e_4)}{P_R(e_1; e_3)}$$

- P_R describes path to the right
  similarly P_L describes path to the left
- Significance is computed based on D_R and D_L wrt parameter
  - Informally: find significant changes in number of paths
- Pick most significant pattern

Constituent-Context Model (CCM)

- Consider all possible binary tree structures on POS sequences
- Define a probability distribution over the possible bracketings
  - A bracketing is a particular structure on a sequence

$$P(s, B) = P_{\text{bin}}(B)P(s|B)$$

$$P(s|B) = \Pi_{i,j:i \leq j} P_{\text{span}}(s_{ij}|B_{ij}) P_{\text{ctx}}(s_{i-1}, s_j|B_{ij})$$

- Run (iterative) Expectation-Maximization (EM) algorithm
  - to maximize likelihood $\Pi_{s \in S} P(s)$
- (Klein 2002)

Dependency Model with Valence (DMV)

- DMV aims to learn dependency relations
  in contrast to CCM which learns context-free grammar rules
- Dependency parse links words in a head-dependent relation
- Model describes likelihood of
  - left dependencies
  - right dependencies
  - stop condition (no more dependencies)
- Again, iterative EM is used to maximize likelihood of corpus

CCM+DMV

- CCM and DMV can be combined
- Both models have different view on structure
- Results of combined system are better than either systems
  - Strengths of both systems are combined
- (Klein 2004)

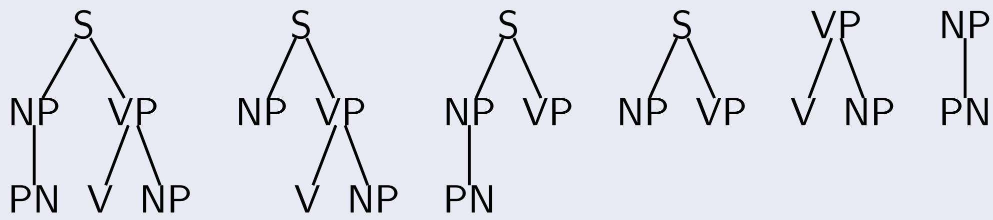
U-DOP

- Similar to CCM in that it
  - finds probability distribution over "all" structures
  - uses POS sequences
- U-DOP uses Data-Oriented Parsing (DOP) as formalism
  - Extends probabilistic model of context-free grammars
- Requires practical implementation choices
  - Random sampling due to huge size of search space
- (Bod 2006a, b)

Procedure

1. Generate all possible binary trees on example sentences
2. Extract all *subtrees*
3. Estimate probabilities on subtrees using EM

Subtrees

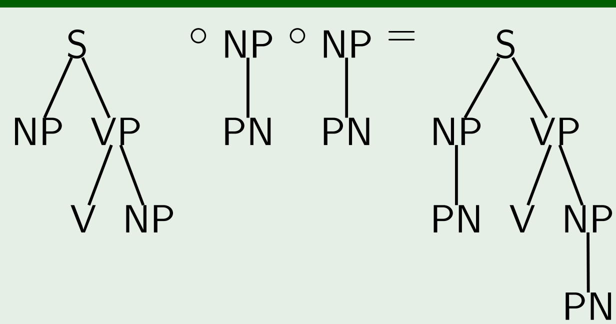


- Remove either all or no elements on a level
  - Leads to *many* subtrees
- Each subtree receives a probability
  - Longer distance dependencies may be modeled

Parsing

- Subtrees can be recombined into a larger tree
  - Similar to context-free grammar rules
- Same parse may be created using different derivations
  - Statistical model has to take this into account

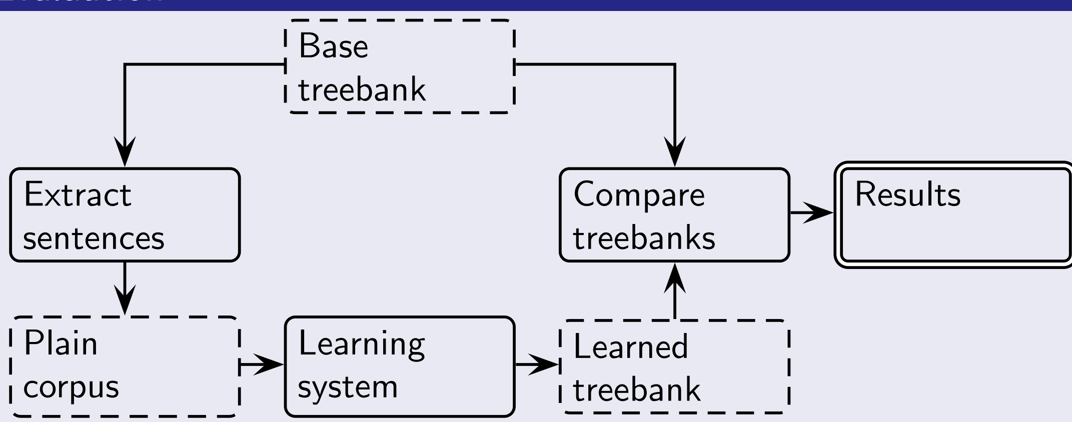
Example



Underlying idea

- U-DOP works because span of subtrees reoccur in a corpus
  - Likelihood of "useful" spans increase
  - Hence, likelihood of contexts (also subtrees) increase
- Essentially, U-DOP uses implied substitutability
  - while system leans heavily on probabilities

Evaluation



- Recall (completeness)
- Precision (correctness)
- F-Score (combination of Precision and Recall)
- (van Zaanen and Adriaans 2001)

Evaluation settings

- Air Travel Information System (ATIS)
- Taken from Penn Treebank II
- 568 English sentences

Example

list the flights from baltimore to seattle that stop in minneapolis
does this flight serve dinner
the flight should arrive at eleven a.m. tomorrow
what airline is this

Results on ATIS

	Micro	Macro	Macro^2
Precision	47.01	46.18	46.18
Recall	44.94	50.98	50.98
F-Score	44.60	47.10	48.46

Explanation

Micro Count constituents, weighted average per sentence
Macro Count constituents and average per sentence
Macro^2 Compute Macro Precision/Recall, average at end

Results on ATIS

		remove sentence	remove empty	remove both
Micro Precision	47.01	47.67	77.10	79.07
Micro Recall	44.94	45.30	44.95	45.29
Micro F-Score	44.60	45.09	55.31	56.13
Macro Precision	46.18	47.66	77.08	81.18
Macro Recall	50.98	52.96	51.07	52.80
Macro F-Score	47.10	48.62	60.00	62.47
Macro^2 F-Score	48.46	50.17	61.43	63.99

Example

(bla bla bla) → bla bla bla
bla () bla → bla bla
(bla () bla) → bla bla

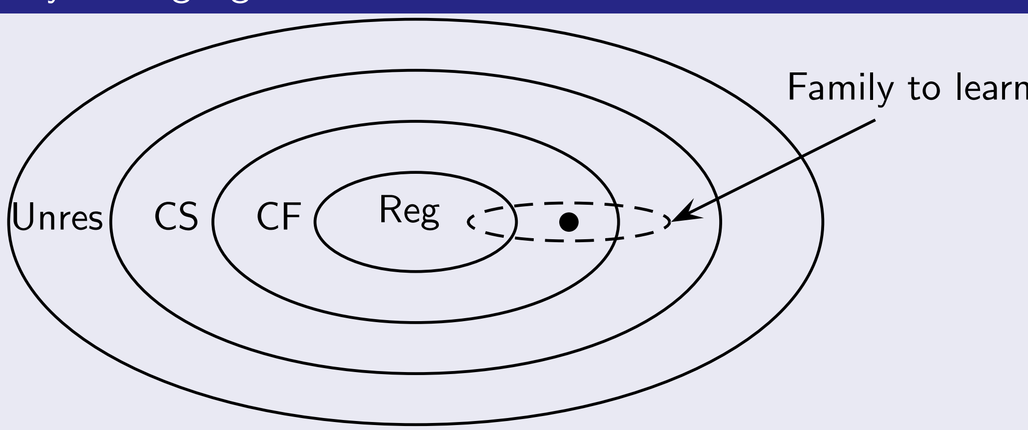
Evaluation insights

- No standard evaluation exists
  but de facto evaluation datasets arise
  - ATIS (van Zaanen and Adriaans 2001)
  - WSJ10, WSJ40 (WSJ with sentence length limitations)
  - NEGRA10 (German)
  - CTB10 (Chinese)
- Systems have different input/output
- Evaluation settings influence results
  - Different metrics (micro/macro/macro^2)
  - Included constituents (sentence/empty)
- Formal grammatical inference does not have this problem
  - Evaluation performed through formal proofs

Context-sensitive grammars

- Learning context-free grammars is hard
- Is learning context-sensitive grammars impossible?
  - That depends
  - To what degree is the grammar context-sensitive?
- We may not need "full" context-sensitiveness
  - Grammar rules: $\alpha A \beta \to \alpha \gamma \beta$
- Mildly context-sensitive grammars may be enough for NL (Huybrechts 1984, Shieber 1985)
- Perhaps the full power of context-freeness is not needed

Family of languages



Learning context-sensitive languages

- Open research area
- Some work has already been done
  - Augmented Regular Expressions (Alquézar 1997)
  - Variants of substitutability (Yoshinaka 2009)
  - Distributional Lattice Grammars (Clark 2010)

Relationship between empirical and formal GI

- Is there a relationship between empirical GI and formal GI?
- Example: consider the case of substitutability
- There are situations in which substitutability breaks:
  - John eats <u>meat</u>
  - John eats <u>much</u>
- This suggests that learning based on substitutability learns a different family of languages (not CFG)
- Non-terminally separated (NTS) languages
  - Subclass of deterministic context-free grammars

Learning NTS grammars

- Grammar G=$\langle \Sigma, V, P, S \rangle$ is NTS
  - Σ is vocabulary
  - V is set of non-terminals
  - P is set of production rules
  - $S \in V$ is the start symbol
- Additional restriction:
  - If $N \in V$
  - $N \stackrel{*}{\Rightarrow} \alpha\beta\gamma$
  - $M \stackrel{*}{\Rightarrow} \beta$
  - then $N \stackrel{*}{\Rightarrow} \alpha M\gamma$
- In other words:
  non-terminals correspond exactly with substitutability
- (Clark and Eyraud 2005, Clark 2006, Clark and Eyraud 2007)

Learning NTS grammars

- It can be shown that NTS grammars are
  - identifiable in the limit
  - PAC learnable
- Unfortunately, natural language is not an NTS language
- Ultimate goal:
  - Find family of languages that fits natural language
  - and is learnable in the right learning setting

Formal GI and empirical GI

- Relation between formal GI and empirical GI
  - Formal GI can show learnability
    - Under certain conditions
  - Emprical GI tries to learn structure from real data
    - Practically shows possibilities and limitations
- Ultimate aim: Find family of languages that is
  - learnable under different conditions
  - fits natural languages

CONCLUSIONS

1. There have been new strong positive results in a recent past for all the cases mentioned (subclasses of regular, PFA, transducers, CFGs, MCSGs)
2. Look for ICGI! It's the conference where these exciting results happen (as well as exciting challenges, competitions, benchmarks etc.)
3. The use of GI techniques both in computational linguistics and natural language processing is taking place.
4. The future is bright!

References

P. W. Adriaans and M. van Zaanen. 2004. Computational grammar induction for linguists. *Grammars*, 7:57–68. Special issue with the theme "Grammar Induction".

P. W. Adriaans and M. van Zaanen. 2006. Computational grammatical inference. In D. E. Holmes and L. C. Jain, editors, *Innovations in Machine Learning*, volume 194 of *Studies in Fuzziness and Soft Computing*, chapter 7. Springer-Verlag, Berlin Heidelberg, Germany. To be published. ISBN: 3-540-30609-9.

P. W. Adriaans and M. Vervoort. 2002. The EMILE 4.1 grammar induction toolbox. In P. W. Adriaans, H. Fernau, and M. van Zaanen, editors, *Grammatical Inference: Algorithms and Applications (ICGI); Amsterdam, the Netherlands*, volume 2482 of *Lecture Notes in AI*, pages 293–295, Berlin Heidelberg, Germany, September 23–25. Springer-Verlag.

P. W. Adriaans. 1992. *Language Learning from a Categorial Perspective*. Ph.D. thesis, University of Amsterdam, Amsterdam, the Netherlands, November.

R. Alquézar and A. Sanfeliu. 1997. Recognition and learning of a class of context-sensitive languages described by augmented regular expressions. *Pattern Recognition*, 30(1):163–182.

D. Angluin and M. Kharitonov. 1991. When won't membership queries help? In *Proceedings of 24th* ACM *Symposium on Theory of Computing*, pages 444–454, New York. ACM Press.

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

D. Angluin. 1982. Inference of reversible languages. *Journal for the Association of Computing Machinery*, 29(3):741–765.

D. Angluin. 1987a. Learning regular sets from queries and counterexamples. *Information and Control*, 39:337–350.

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

D. Angluin. 1988. Identifying languages from stochastic examples. Technical Report YALEU/DCS/RR-614, Yale University, March.

R. B. Applegate. 1972. *Ineseño Chumash Grammar*. Ph.D. thesis, University of California, Berkeley.

L. Beccera-Bonache, C. Bibire, and A. Horia Dediu. 2005. Learning DFA from corrections. In Henning Fernau, editor, *Proceedings of the Workshop on Theoretical Aspects of Grammar Induction (*TAGI*)*, WSI-2005-14, pages 1–11. Technical Report, University of Tübingen.

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

D. Béchet, A. Dikovsky, and A. Forêt. 2011. Sur les itérations dispersées et les choix itŕés pour l'apprentissage incrémental des types dans les grammaires de dépendances. In *Proceedings of Conférence d'Apprentissage*.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. 1989. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965.

R. Bod. 1998. *Beyond Grammar—An Experience-Based Theory of Language*, volume 88 of *CSLI Lecture Notes*. Center for Study of Language and Information (CSLI) Publications, Stanford:CA, USA.

R. Bod. 2006a. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING) and 44th Annual Meeting of the Association of Computational Linguistics (ACL); Sydney, Australia*, pages 865–872. Association for Computational Linguistics.

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

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

E. Charniak. 1993. *Statistical Language Learning*. Massachusetts Institute of Technology Press, Cambridge:MA, USA and London, UK.

N. Chater and P. Vitányi. 2007. 'ideal learning' of natural language: Positive results about learning from positive evidence. *Journal of Mathematical Psychology*, 51(3):135–163.

N. Chomsky. 1957. *Syntactic Structures*. Mouton & Co., Printers, The Hague.

A. Clark and R. Eyraud. 2005. Identification in the limit of substitutable context-free languages. In S. Jain, H. U. Simon, and E. Tomita, editors, *Algorithmic Learning Theory: 16th International Conference, ALT 2005*, volume 3734 of *Lecture Notes in Computer Science*, pages 283–296, Berlin Heidelberg, Germany. Springer-Verlag.

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

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

A. Clark. 2006. PAC-learning unambiguous NTS languages. In Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, and E. Tomita, editors, *Eighth International Colloquium on Grammatical Inference, (ICGI); Tokyo, Japan*, number 4201 in Lecture Notes in AI, pages 59–71, Berlin Heidelberg, Germany. Springer-Verlag.

A. Clark. 2010. Efficient, correct, unsupervised learning of context-sensitive languages. In *CoNLL '10: Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 28–37, Stroudsburg, PA, USA. Association for Computational Linguistics.

C. de la Higuera and J. Oncina. 2004. Learning probabilistic finite automata. In G. Paliouras and Y. Sakakibara, editors, *Grammatical Inference: Algorithms and Applications, Proceedings of* ICGI *'04*, volume 3264 of LNAI, pages 175–186. Springer-Verlag.

C. de la Higuera and F. Thollard. 2000. Identification in the limit with probability one of stochastic deterministic finite automata. In A.L. de Oliveira, editor, *Grammatical Inference: Algorithms and Applications, Proceedings of ICGI '00*, volume 1891 of *Lecture Notes in Computer Science*, pages 15–24. Springer-Verlag.

C. de la Higuera, J.-C. Janodet, and F. Tantini. 2008. Learning languages from bounded resources: the case of the DFA and the balls of strings. In A. Clark, F. Coste, and L. Miclet, editors, *Grammatical Inference: Algorithms and Applications, Proceedings of* ICGI *'08*, volume 5278 of LNCS, pages 43–56. Springer-Verlag.

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

C. de la Higuera. 2010. *Grammatical inference: learning automata and grammars*. Cambridge University Press, Cambridge, UK.

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

Matt Edlefsen, Dylan Leeman, Nathan Myers, Nathaniel Smith, Molly Visscher, and David Wellcome. 2008. Deciding strictly local (SL) languages. In Jon Breitenbucher, editor, *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pages 66–73.

Jie Fu, J. Heinz, and Herbert Tanner. 2011. An algebraic characterization of strictly piecewise languages. In *The 8th Annual Conference on Theory and Applications of Models of Computation*, volume 6648 of *Lecture Notes in Computer Science*. Springer-Verlag.

P. García and J. Ruiz. 2004. Learning k-testable and k-piecewise testable languages from positive data. *Grammars*, 7:125–140.

P. Garcia and E. Vidal. 1990. Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:920–925.

P. Garcia, E. Vidal, and J. Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, pages 325–338.

G.Clements and J. Keyser. 1983. *CV phonology: a generative theory of the syllable*. Cambridge, MA: MIT Press.

J. Geertzen and M. van Zaanen. 2004. Grammatical inference using suffix trees. In G. Paliouras and Y. Sakakibara, editors, *Grammatical Inference: Algorithms and Applications: Seventh International Colloquium, (ICGI); Athens, Greece*, volume 3264 of *Lecture Notes in AI*, pages 163–174, Berlin Heidelberg, Germany, October 11–13. Springer-Verlag.

D. Gildea and D. Jurafsky. 1996. Learning bias and phonological-rule induction. *Computational Linguistics*, 24(4).

L. Gleitman. 1990. The structural sources of verb meanings. *Language Acquisition*, 1(1):3–55.

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

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

K. C. Hansen and L. E. Hansen. 1969. Pintupi phonology. *Oceanic Linguistics*, 8:153–170.

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

B. Hayes. 1995. *Metrical Stress Theory*. Chicago University Press.

J. Heinz and J. Rogers. 2010. Estimating strictly piecewise distributions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 886–896, Uppsala, Sweden, July. Association for Computational Linguistics.

J. Heinz. 2008. Left-to-right and right-to-left iterative languages. In Alexander Clark, François Coste, and Lauren Miclet, editors, *Grammatical Inference: Algorithms and Applications, 9th International Colloquium*, volume 5278 of *Lecture Notes in Computer Science*, pages 84–97. Springer.

J. Heinz. 2009. On the role of locality in learning stress patterns. *Phonology*, 26(2):303–351.

J. Heinz. 2010. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 897–906, Uppsala, Sweden, July. Association for Computational Linguistics.

R. M. A. C. Huybrechts. 1984. The weak adequacy of context-free phrase structure grammar. In G. J. de Haan, M. Trommelen, and W. Zonneveld, editors, *Van periferie naar kern*, pages 81–99. Foris, Dordrecht, the Netherlands.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, December.

A. Kasprzik and T. Kötzing. 2010. String extension learning using lattices. In Henning Fernau Adrian-Horia Dediu and Carlos Martín-Vide, editors, *Proceedings of the 4th International Conference on Language and Automata Theory and Applications (LATA 2010)*, volume 6031 of *Lecture Notes in Computer Science*, pages 380–391, Trier, Germany. Springer.

M. Kearns and L. Valiant. 1989. Cryptographic limitations on learning boolean formulae and finite automata. In *21st* ACM *Symposium on Theory of Computing*, pages 433–444.

M. J. Kearns and U. Vazirani. 1994. *An Introduction to Computational Learning Theory*. MIT press.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *40th Annual Meeting of the Association for Computational Linguistics; Philadelphia:PA, USA*, pages 128–135. Association for Computational Linguistics, July. yes.

D. Klein. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *42th Annual Meeting of the Association for Computational Linguistics; Barcelona, Spain*, pages 479–486.

G. Kobele. 2006. *Generating Copies: An Investigation into Structural Identity in Language and Grammar*. Ph.D. thesis, University of California, Los Angeles.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4(35–56).

R. McNaughton and S. Papert. 1971. *Counter-Free Automata*. MIT Press.

M. Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

S. Muggleton. 1990. *Inductive Acquisition of Expert Knowledge*. Addison-Wesley.

J. Oncina and P. García. 1992. Identifying regular languages in polynomial time. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 99–108. World Scientific.

J. Oncina, P. García, and E. Vidal. 1993. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, May.

L. Pitt. 1985. *Probabilistic Inductive Inference*. Ph.D. thesis, Yale University. Computer Science Department, TR-400.

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

J. Rogers and G. Pullum. to appear. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*.

J. Rogers, J. Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In Christian Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*, volume 6149 of *Lecture Notes in Artifical Intelligence*, pages 255–265. Springer.

S. M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343.

I. Simon. 1975. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222.

Z. Solan, D. Horn, E. Ruppin, and S. Edelman. 2005. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634, August.

A. Stolcke. 1994. *Bayesian Learning of Probabilistic Language Models*. Ph.D. thesis, University of California, Berkeley.

I. Tellier. 2008. How to split recursive automata. In *ICGI*, pages 200–212.

E. Ukkonen. 1995. On-line construction of suffix trees. *Algorithmica*, 14:249–260.

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

M. van Zaanen and P. W. Adriaans. 2001. Alignment-Based Learning versus EMILE: A comparison. In *Proceedings of the Belgian-Dutch Conference on Artificial Intelligence (BNAIC); Amsterdam, the Netherlands*, pages 315–322, October.

M. van Zaanen. 2000a. ABL: Alignment-Based Learning. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING); Saarbrücken, Germany*, pages 961–967. Association for Computational Linguistics, July 31–August 4.

M. van Zaanen. 2000b. Bootstrapping syntax and recursion using Alignment-Based Learning. In P. Langley, editor, *Proceedings of the Seventeenth International*

Conference on Machine Learning; Stanford:CA, USA, pages 1063–1070, June 29–July 2.

M. van Zaanen. 2002. *Bootstrapping Structure into Language: Alignment-Based Learning*. Ph.D. thesis, University of Leeds, Leeds, UK, January.

Marco R. Vervoort. 2000. *Games, Walks and Grammars*. Ph.D. thesis, University of Amsterdam, Amsterdam, the Netherlands, September.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. 2005a. Probabilistic finite-state machines-part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. 2005b. Probabilistic finite-state machines-part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1026–1039.

R. A. Wagner and M. J. Fischer. 1974. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.

R. Wiehagen, R. Frievalds, and E. Kinber. 1984. On the power of probabilistic strategies in inductive inference. *Theoretical Computer Science*, 28:111–133.

T. Yokomori. 2003. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 298(1):179–206.

R. Yoshinaka. 2009. Learning mildly context-sensitive languages with multidimensional substitutability from positive data. In R. Gavaldà, G. Lugosi, T. Zeugmann, and S. Zilles, editors, *Proceedings of the Workshop on Algorithmic Learning Theory*, volume 5809 of *Lecture Notes in Computer Science*, pages 278–292. Springer Berlin / Heidelberg.