Distributional Learning

Learnability and Language Acquisition

Alexander Clark

Department of Computer Science
Royal Holloway, University of London

Thursday

Outline

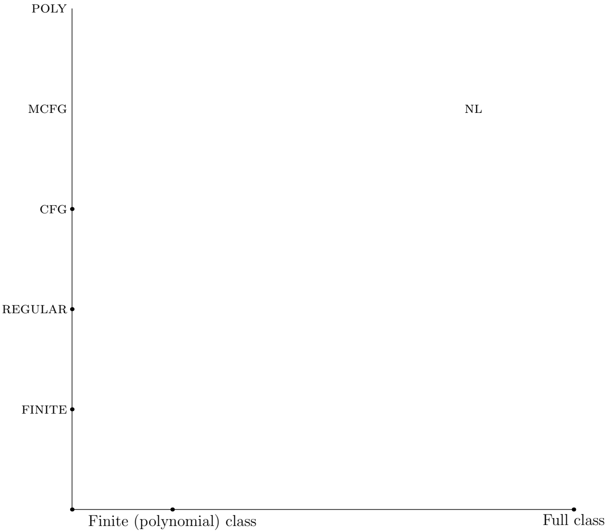
Provably Learnable Language Classes

- We are focusing here on algorithms that can be proved to learn a class of languages according to certain criteria.

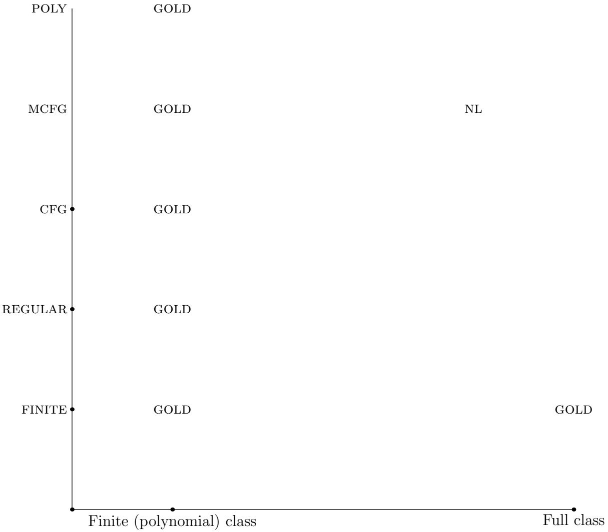
Provably Learnable Language Classes

- We are focusing here on algorithms that can be proved to learn a class of languages according to certain criteria.
- These procedures differ from heuristic procedures, which yield experimental results for grammar induction from a naturally occurring corpus, but are not necessarily provably correct for an entire class of languages or representations.
- Significant progress has been made in recent years on the development of both types of grammar induction algorithms.

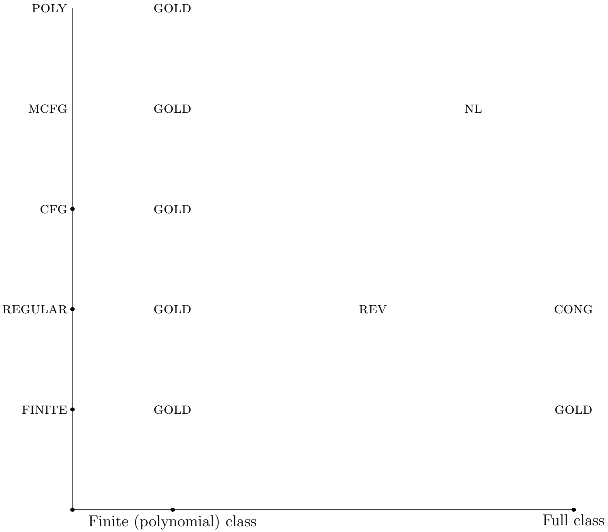
The class of natural languages



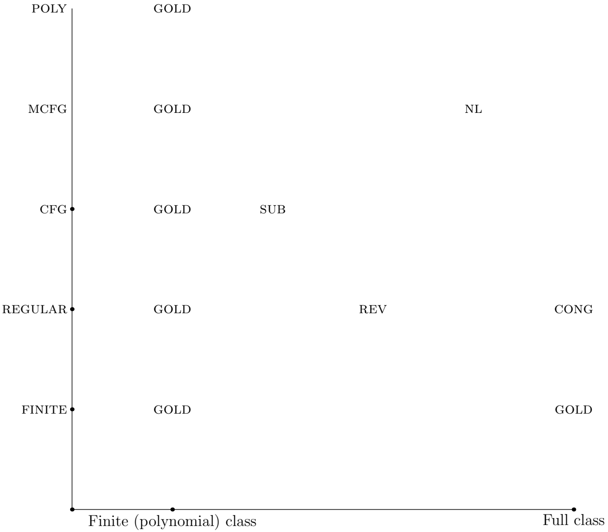
1967


POLY
GOLD
MCFG
GOLD
NL
CFG
GOLD
REGULAR
GOLD
FINITE
GOLD
GOLD
Finite (polynomial) class
Full class

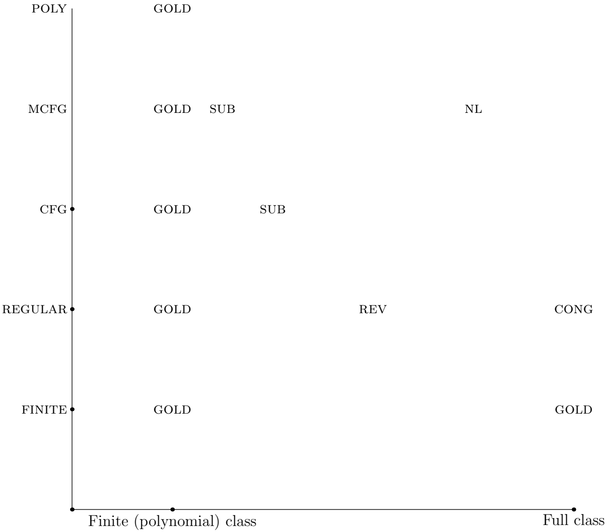
1987


POLY
GOLD
MCFG
GOLD
NL
CFG
GOLD
REGULAR
GOLD
REV
CONG
FINITE
GOLD
GOLD
Finite (polynomial) class
Full class

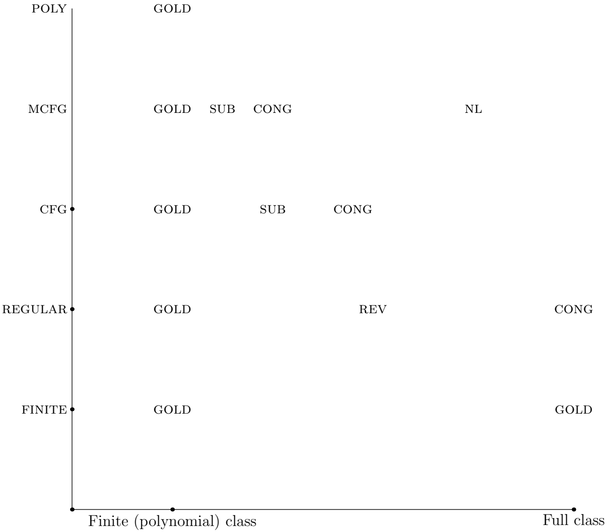
2005


POLY
MCFG
CFG
REGULAR
FINITE
GOLD
GOLD
GOLD
GOLD
GOLD
NL
SUB
REV
CONG
GOLD
Finite (polynomial) class
Full class

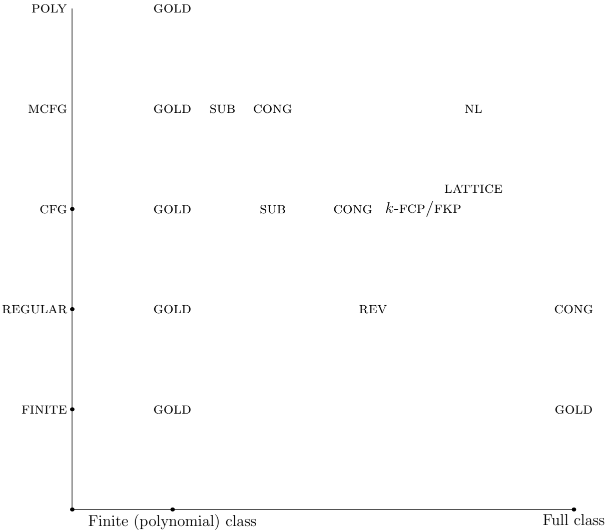
2008


POLY
MCFG
CFG
REGULAR
FINITE
GOLD
GOLD
SUB
NL
GOLD
SUB
GOLD
REV
CONG
GOLD
GOLD
Finite (polynomial) class
Full class

2010


POLY
GOLD
MCFG
GOLD SUB CONG
NL
CFG
GOLD SUB CONG
REGULAR
GOLD REV CONG
FINITE
GOLD GOLD
Finite (polynomial) class
Full class

2011


POLY
MCFG
CFG
REGULAR
FINITE
GOLD
GOLD SUB CONG NL
LATTICE
GOLD SUB CONG k-FCP/FKP
GOLD REV CONG
GOLD GOLD
Finite (polynomial) class
Full class

Weaknesses of these approaches

- Classes of languages are (mostly) too small
- Learning models are (mostly) too easy and/or too idealised
- They (mostly) lack an appropriate “feature calculus”
- These are (all) just weak learnability results.

Outline

American structuralism

Structuralist tradition

- Bloomfield
- Rulon Wells
- Zellig Harris

Harris, 1940

Reviewing a nonstructuralist book

the value of the book is vitiated, especially for the layman, by a major short- coming. This is the neglect of the method of structural analysis, i.e. of organized synchronic description. As a result, many of the facts about languages are misconstrued, and linguistic theory is distorted. It is the chief purpose of this review to show that **an appreciation of linguistic structure is necessary for any interpretation of linguistics**, and that its neglect leads to undesirable results in practice.

Rejecting historical approach and its focus on written language.

And its rejection . . .

Chomsky, 1965

The only proposals that are explicit enough to support serious study are those that have been developed within taxonomic linguistics. It seems to have been demonstrated beyond reasonable doubt that quite apart from any questions of feasibility, methods of the sort that have been studied in taxonomic linguistics are intrinsically incapable of yielding the systems of grammatical knowledge that must be attributed to the speaker of a language.

And its rejection . . .

Chomsky, p.c.

From the 50s, there has seemed to me no hope in distributional procedures.

Why was it rejected?

- Distributional learning was perceived as a discovery procedure
- No mathematically precise models (contra Harris)
- Problems of complexity unless the range of variation is finite
- A sequence of phrase markers – unlearnable because you only observe the last one (Katz and Postal, 1964)
- No way of dealing with structure-dependent movement
- No model of ambiguity or of syntactic structure

Distinction

Discovery procedure

Used by linguists to automatically generate a grammar
But if a grammar is a theory, why do we need to automatically generate it?

Model of language acquisition

Rather than a linguist analyzing a corpus, we have a child processing the primary linguistic data

Kulagina school

Structuralist linguistics died out in America after Chomsky.

Oettinger, 1958

The latter paper (Kulagina 1957) has considerable expository merit, and it is clearer and more sensible than simi- lar papers on set-theoretic concepts in language which have sprouted like ungainly weeds in the lawn of our information-retrieval literature. The work is along somewhat different lines, and of lesser extent but of caliber comparable to that of the excellent theoretical work of Chomsky in this country.

Solomon Marcus, Sestier, Kunze, . . .

Direct psycholinguistic evidence

Artificial Grammar Learning in infants

Saffran, Aslin, Newport (1996) . . .

Human simulation experiments

Gillette, J. et al. (1999), Gleitman (1990), . . .

Lexical acquisition experiments

Mintz, T. (2002), Childers and Tomasello (2001), . . .

Children and adults do exploit distributional evidence.

Computational experiments

Natural language processing

Brown et al. (1992), Curran, J. (2003), . . .
Standard components of large NLP systems.

CHILDES Experiments

Redington, Fitch, & Chater (1998), Mintz, T. (2003), . . .

These experiments show that rich evidence is available in reasonably sized natural corpora.

Distributional learning

Chomsky (1968/2006)

“The concept of "phrase structure grammar" was explicitly designed to express the richest system that could reasonable be expected to result from the application of Harris-type procedures to a corpus.”

Outline

Regular languages

A natural class (not like context free languages)

- Language theoretic characterisation (Myhill-Nerode theorem)
- Two machine models (DFA, NFA)
- A grammar model (Regular grammars)

Regular languages

A natural class (not like context free languages)

- Language theoretic characterisation (Myhill-Nerode theorem)
- Two machine models (DFA, NFA)
- A grammar model (Regular grammars)

- Predate the Chomsky hierarchy and formal language theory.
- The “natural numbers” of languages (Dedekind)

Regular inference

A fairly complete theory of learnability

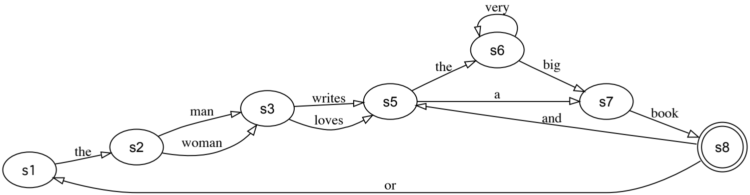
Paradigm	Learnable class	
Positive Data	reversible languages	Angluin (1982)
Queries	regular languages	Angluin (1987)
Positive and Negative	regular languages	Oncina and Garcia (1992)
Stochastic data	acyclic PDFAs	Ron et al (1994),
	regular languages	Carrasco and Oncina (1994
	regular languages	Clark and Thollard (2004)

In practice, random DFAs are learnable from positive data alone.

Regular languages

- If natural languages were in fact regular, then the debate would be over.
- But they aren't right for natural languages:
  - natural languages are not weakly regular
  - natural languages have some non-regular structure
- Important though to show that the APS arguments are wrong.

Representation is a DFA



Representational assumption

Look at relation between prefix and suffix

String “the man loves the big book” is in the language this means that

- “the” can occur before “man loves the big book”
- “the man ” can occur before “loves the big book”
- “the man loves” can occur before “ the big book”

Learning

Data

the man loves the big book
the man writes a book
the woman loves the big book and a book
the woman writes a book
. . .

Learning

Data

the man loves the big book
the man writes a book
the woman loves the big book and a book
the woman writes a book
...

Pick a prefix “the man”

loves the big book
writes a book

Pick a prefix “the woman”

loves the big book and a book
writes a book

The suffix sets look similar ...

Big idea

Equivalence of sets of suffixes

- If two strings end up in the same state, then they will have the same set of suffixes.
- If two strings have the same set of suffixes, then they end up in the same state.
- Pick the states to be sets of prefixes that have the same set of suffixes

Big idea

Equivalence of sets of suffixes

- If two strings end up in the same state, then they will have the same set of suffixes.
- If two strings have the same set of suffixes, then they end up in the same state.
- Pick the states to be sets of prefixes that have the same set of suffixes

Problem

How to tell whether two strings have the same set of suffixes?

Testing

How to tell when two strings are equivalent?

Three approaches:

- Restrict the class of languages so it is easy to tell from positive examples alone, even when generated from an adversary.
- Allow some other form of queries so we can test.
- Make some assumptions about the distribution of examples, so we can test probabilistically.

Reversibility

Angluin 1982

Definition

If $uv, u'v, uv' \in L$ then $u'v' \in L$

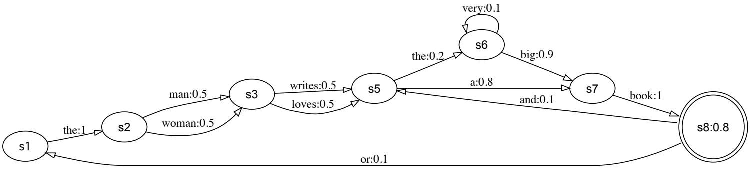
Informally

If two strings have one suffix in common, then they have all suffixes in common.

Reversible regular languages

IIL from positive data alone
Restricted subclass that doesn't include all finite languages
e.g. $\{a, aa\}$ is not reversible

A probabilistic DFA



Remarks

- This defines a probability distribution over all strings.
- Strings not accepted by the automaton get probability zero.
- Strings accepted by the automaton get positive probability – since all of the parameters are non-zero.
- We have an infinite family of distributions.
- There are many other distributions that can't be described by a PDFA.

Comparison

Compare distribution starting from s5 and from s7

Suffix	s5	s7	difference
a book	0.64	0	0.64
the big book	0.144	0	0.144
the very big book	0.0144	0	0.0144
book	0	0.8	0.8
book and a book	0	0.064	0.064

Definition

Distinguishability between these two states is the maximum difference $= 0.8$

Distinguishability of a PDFA

- Learning PDFAs is computationally hard (Kearns et al., 1994)
- Hard PDFAs have two states that are very hard to distinguish – i.e. with very low distinguishability.
- Define distinguishability of a PDFA (μ) is the minimum distinguishability between states in a PDFA.
  - PDFAs with very small μ are hard to learn
  - PDFAs with large μ are easy to learn

Two Learnability Results for PDFAs

Clark and Thollard

PDFAs are learnable as distributions

We can PAC-learn PDFAs in polynomial data and computation from positive examples when the sample complexity depends on n, μ , $|\Sigma|$, and D a bound on the expected length of strings from any state.

DFAs are learnable when the data is generated by a PDFA

We can PAC-learn DFAs when the samples are generated by a PDFA that defines the same language, when the sample complexity depends on n, μ and $|\Sigma|$

Limitations of PDFA

- PDFAs are efficiently learnable from positive data because their states and transitions are easily identifiable from observable linguistic evidence.
- PDFAs generate the set of regular languages, some of which are infinite.
- They can capture important syntactic and morphological features of natural languages.
- But they lack the expressive power to represent the full range of syntactic structures in natural language, some of which exhibit context free, or even mildly context sensitive properties.

Learning model

Probabilistic data

Learn the whole class of PDFAs
learn the whole class of DFAs when data is drawn from a suitable distribution

Query model

Given membership queries (Is *w* in the target language?)
Learn the whole class of DFAs (Angluin, 1987)

Conclusion/Conjecture

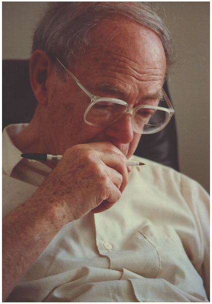
Learning with reasonable distributions is (roughly) equivalent to learning when you have membership queries.
This is close to how difficult it is to learn in practice.

Outline

Distributional Learning

Zellig Harris (1949, 1951)

Here as throughout these procedures X and Y are substitutable if for every utterance which includes X we can find (or gain native acceptance for) an utterance which is identical except for having Y in the place of X



Example

Is ’cat’ substitutable for ’dog’?

- The cat is over there.
- I want a dog for Christmas.
- I want a Siamese cat for Christmas.
- Put a cat-flap in the door to the kitchen.
- An Alsatian is a breed of dog.
- He continues to dog my footsteps.
- I would rather have a dog than a cat as a pet.

Example

Is ’cat’ substitutable for ’dog’?

- The dog is over there.
- I want a cat for Christmas.
- I want a Siamese dog for Christmas.
- Put a dog-flap in the door to the kitchen.
- An Alsatian is a breed of cat.
- He continues to cat my footsteps.
- I would rather have a dog than a dog as a pet.
- I would rather have a cat than a dog as a pet.

Empirical work on Distributional Learning

Real corpora

- Sample is not just of grammatical sentences
- Also semantically well-formed
- Also "true" in some non-technical sense
- Empirical distribution is very complex

Distributional similarity in real corpora often reflects semantic relatedness

Various notions of context

The word 'has' in the sentence: "If the candidate has an outstanding examination result"

Local syntactic context

Immediately preceding and following word
(candidate, an)

Wide bag-of-words context

Skip stop words
Set of words occurring in the same sentence/discourse.
{ candidate, examination, outstanding, result }

Various notions of context

The word ’has’ in the sentence: “If the candidate has an outstanding examination result”

Local syntactic context

Immediately preceding and following word
(candidate, an)

Wide bag-of-words context

Skip stop words
Set of words occurring in the same sentence/discourse.
{ candidate, examination, outstanding, result }

Full context

“If the candidate _ an outstanding examination result”

Example

Distribution of “cat” in English

Infinite set of full contexts that ’cat’ can appear in :
“the _ is over there”
“I want a _ for Christmas”
. . .

- We can observe the distribution simply by looking at positive examples.
- We can see a similarity in distribution of “cat” and “dog”.
- Distributional learning is based on this idea.

Distribution

Full context

Context (or *environment*)

A context is just a pair of strings $(l, r) \in \Sigma^* \times \Sigma^*$.

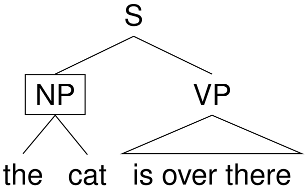
Context substring relation

$(l, r) \sim u$ if $lur \in L$
Special context (λ, λ)

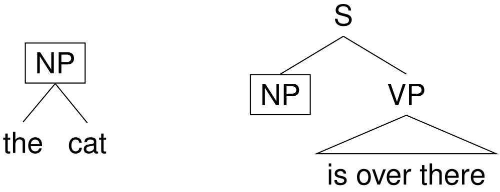
Distribution of a string

Given a language L
$C_L(u) = \{(l, r) | lur \in L\}$

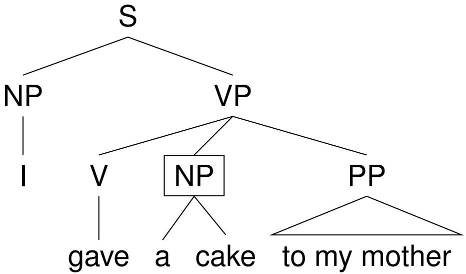
Trees



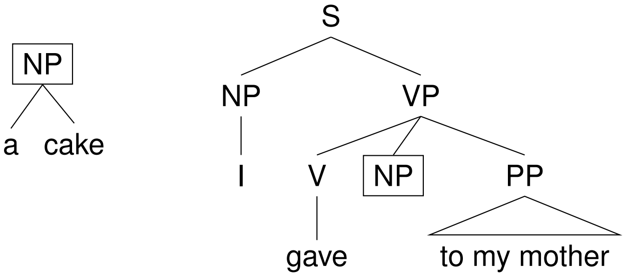
Trees



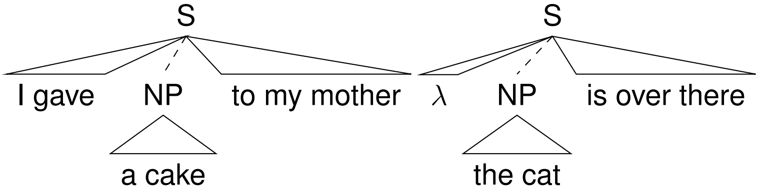
Trees



Trees



Contexts



Contexts and yields of nonterminals

Yield of a non-terminal

$Y_G(NP)$ is the set of all strings w such that $NP \stackrel{*}{\Rightarrow} w$

$Y_G(NP) = \{$ the cat, the dog, some blue boxes $\ldots \}$

Contexts of a non-terminal

$C_G(NP)$ is the set of all contexts (l, r) such that $S \stackrel{*}{\Rightarrow} lNPr$

$C_G(NP) = \{$ _ is over there, I want _, Put it on _ $\ldots \}$

Any string in $Y(NP)$ can occur in any context in $C(NP)$

A difficult question

Suppose we have some string, say 'the cat', which is in $Y(NP)$

Question

What is the relationship between the distribution of 'the cat'
$C_L(\text{the cat})$
and the contexts or distribution of NP: $C_G(NP)$??

Congruence classes

Congruence classes

$u \equiv v$ iff $C_L(u) = C_L(v)$
Write $[u]$ for class of u
This is the set of all strings that are perfectly substitutable for each other in every context.

- **Equality of distribution is the same as complete substitutability**
- Two strings are congruent if they are perfectly substitutable in every context.
- Words: Tuesday/Wednesday, cat/dog, man/student
- In formal languages, this is a more useful idea
- This gives us a partition of every substring into classes

Congruence classes of a simple formal language

Simple finite language

It is enormous. It is big.
He is enormous. He is big.

Congruence classes of a simple formal language

Simple finite language

It is enormous. It is big.
He is enormous. He is big.

- It, he
- big, enormous
- is

Congruence classes of a simple formal language

Simple finite language

It is enormous. It is big.
He is enormous. He is big.

- It, he
- big, enormous
- is
- It is, he is
- is big, is enormous
- L

Congruence classes of a simple formal language

Simple finite language

It is enormous. It is big.
He is enormous. He is big.

- It, he
- big, enormous
- is
- It is, he is
- is big, is enormous
- L
- is is, is he, . . .

Congruence classes

- Suppose ’red’ is congruent to ’blue’ and ’box’ is congruent to ’jug’
- Then ’red box’ is congruent to ’blue jug’

Congruence classes

- Suppose 'red' is congruent to 'blue' and 'box' is congruent to 'jug'
- Then 'red box' is congruent to 'blue jug'

Congruence classes have nice properties!

If $u \equiv u'$ and $v \equiv v'$ then $uv \equiv u'v'$

$[u][v] \subseteq [uv]$

Context free grammar

Rules

$S \rightarrow NP\ VP$
$NP \rightarrow Det\ N$
$Det \rightarrow the$, $N \rightarrow cat$

Yield of a non-terminal

$Y(NP)$ is the set of all strings w such that $NP \stackrel{*}{\Rightarrow} w$
$Y(NP) = \{$ the cat, the dog, some blue boxes $\ldots \}$

Context free grammar

Suppose we have a grammar with non-terminals N, P, Q

- We have a rule $N \rightarrow PQ$
- This means that $Y(N) \supseteq Y(P)Y(Q)$.

Context free grammar

Suppose we have a grammar with non-terminals N, P, Q

- We have a rule $N \to PQ$
- This means that $Y(N) \supseteq Y(P)Y(Q)$.

Backwards

Given a collection of sets of strings X, Y, Z
Suppose $X \supseteq YZ$
Then we add a rule $X \to YZ$.

Basic representational assumption

Representation

Non-terminals correspond to congruence classes.

Basic representational assumption

Representation

Non-terminals correspond to congruence classes.

Congruence classes have nice properties!

$[u][v] \subseteq [uv]$
This means we can always have a rule $[uv] \rightarrow [u][v]$

Congruence classes based rules

$$L = \{ab^n c^n d | n \geq 0\}$$

Trivial classes

$[a] = \{a\}$
$[b] = \{b\}$
$[c] = \{c\}$
$[d] = \{d\}$
$[ab] = \{ab\}$
...

Trivial rule

$[bc] \rightarrow [b][c]$

Interesting classes

$[bc] = \{bc, bbcc, \dots\}$
$[bbc] = \{bbc, bbbcc, \dots\}$
$[bcc] = \{bcc, bbccc, \dots\}$
$[abc] = \{a, abc, abbcc \dots\}$
$[abcd] = L$

Congruence classes based rules

$L = \{ab^n c^n d | n \geq 0\}$

Trivial classes

$[a] = \{a\}$
$[b] = \{b\}$
$[c] = \{c\}$
$[d] = \{d\}$
$[ab] = \{ab\}$
$\ldots$

Interesting classes

$[bc] = \{bc, bbcc, \ldots\}$
$[bbc] = \{bbc, bbbcc, \ldots\}$
$[bcc] = \{bcc, bbccc, \ldots\}$
$[abc] = \{a, abc, abbcc \ldots\}$
$[abcd] = L$

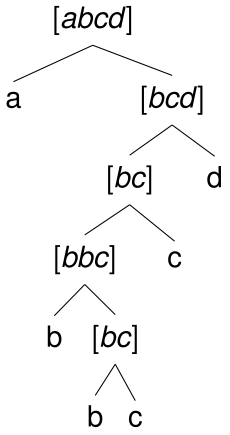
Rule 2

$[bc] = [bbcc]$
$[bbcc] \rightarrow [bbc][c]$
$[bc] \rightarrow [bbc][c]$

Rule 3

$[bbc] \rightarrow [b][bc]$
$[bc] \stackrel{*}{\Rightarrow} [b][bc][c]$

Tree



Constructing CFG from congruence classes

One non-terminal per congruence class

- $[uv] \to [u][v]$
- $[a] \to a$
- $[\lambda] \to \lambda$
- $I = \{[u] | [u] \subseteq L\}$

Multiple start symbols in the CFG doesn't change anything

Two problems

Problem A

- In general we will have an infinite set of congruence classes
- Pick some sufficiently large finite subset of them

Two problems

Problem A

- In general we will have an infinite set of congruence classes
- Pick some sufficiently large finite subset of them

Problem B

Hard to tell whether $u \equiv_L v$
We need some way of testing this.

Outline

Four algorithms

All based on the same representational idea: non-terminals are congruence classes.

1. Substitutable languages from positive data (Clark and Eyraud, 2005, 2007)
2. Congruential languages with queries (Clark, 2010)
3. NTS languages with stochastic positive examples (Clark, 2006)
4. Congruential languages from positive and negative examples (to appear)

Exactly parallel to the results for regular languages.

Old idea 1

Chomsky review of Greenberg, 1959

let us say that two units A and B are substitutable$_1$ if there are expressions X and Y such that XAY and XBY are sentences of L.; substitutable$_2$ if whenever XAY is a sentence of L then so is XBY and whenever XBY is a sentence of L so is XAY (i.e. A and B are completely mutually substitutable). These are the simplest and most basic notions.

Problem:

we need substitutability$_2$ but what we observe is substitutability$_1$

Old idea 2

John Myhill, 1950 commenting on Bar-Hillel

I shall call a system *regular* if the following holds for all expressions μ, ν and all wffs ϕ, ψ each of which contains an occurrence of ν: If the result of writing μ for some occurrence of ν in ϕ is a wff, so is the result of writing μ for any occurrence of ν in ψ. Nearly all formal systems so far constructed are regular; ordinary word-languages are conspicuously not so.

Clark and Eyraud, 2005/2007

A language is *substitutable* if $lur, lvr, l'ur' \in L$ means that $l'vr' \in L$.

substitutable and reversible

Clark and Eyraud, 2005

A language is *substitutable* if $lur, lvr, l'ur' \in L$ means that $l'vr' \in L$.

Angluin, 1982

A language is *reversible* if $ur, vr, ur' \in L$ means that $vr' \in L$.

A Bad Intuition

One context in common in enough

- The cat died
- The dog died

So “cat” and “dog” are congruent

A Bad Intuition

One context in common in enough

- The cat died
- The dog died

So “cat” and “dog” are congruent

- He is an Englishman
- He is thin

So “thin” and “an Englishman” are congruent.

Chomsky example

John is eager to please
John is easy to please

Result

Clark and Eyraud, 2005/2007

Polynomial result

The class of substitutable context free languages is polynomially identifiable in the limit from positive data only.

- Polynomial characteristic set
- Polynomial update time

Why the delay?

A Simple Algorithm

Non-technical description

- Given a sample of strings $W = \{w_1, \ldots, w_n\}$.
- Define a graph $G = \langle N, E \rangle$
  - N is the set of all non-empty substrings (factors) of W.
  - $E = \{(u, v) | \exists (l, r), lur \in W \wedge lvr \in W\}$.
- Define a grammar in Chomsky normal form
  - The set of non-terminals is the set of components of the graph G
  - Have productions for: $[a] \rightarrow a$
  - Add rules for non terminals: $[w] \rightarrow [u][v]$ iff $[w] = [uv]$.

Simple linguistically motivated example

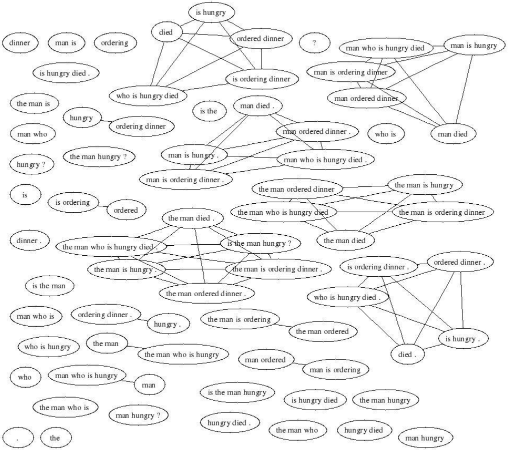
the man who is hungry died .
the man ordered dinner .
the man died .
the man is hungry .
is the man hungry ?
the man is ordering dinner .

---

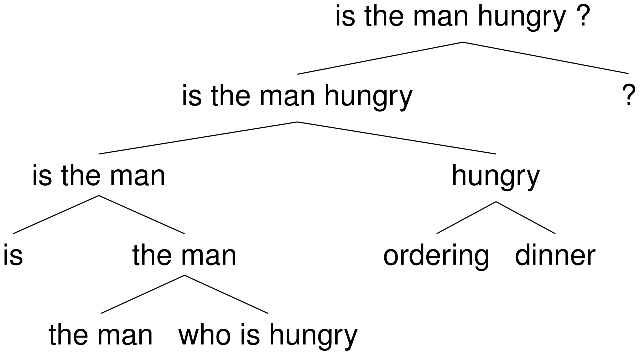
is the man who is hungry ordering dinner ?
$*$is the man who hungry is ordering dinner ?

Substitution graph

Auxiliary fronting example



Tree


is the man hungry ?
is the man hungry
?
is the man
hungry
is
the man
ordering
dinner
the man
who is hungry

Counterexample

Berwick, Coen and Niyogi, p.c

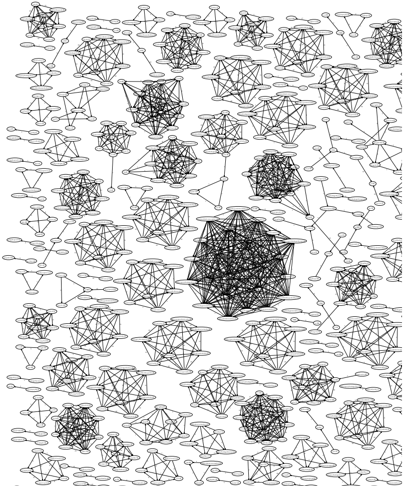
is (bob) well ?
is (the man john) well ?
does he think (well) ?
does he think (hitting is nice) ?
is (bob) (well) ?

---

is the man john hitting is nice ?

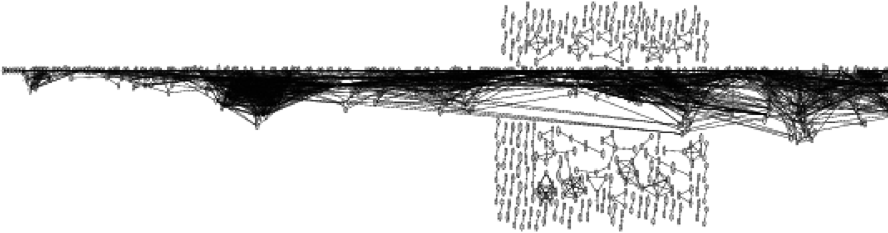
Substitution graph

Learnable example



Substitution graph

Unlearnable languages



Substitutable languages

- Some very basic languages are not substitutable:
  - $L = \{a, aa\}$
  - $L = \{a^n b^n | n > 0\}$
  - Dyck language
- The very strict requirement for contexts to be disjoint is unrealistic.
- The test for congruence is way too weak.
- If we move to a probabilistic learning approach, or queries we can have a better test
- If the test is good, the hypothesis will never overgenerate

Congruence class results

Positive data alone

$lur \in L$ and $lvr \in L$ implies $u \equiv_L v$
Polynomial result from positive data. (Clark and Eyraud, 2005)
k-l substitutable languages, (Yoshinaka 2008)

Stochastic data

If data is generated from a PCFG
PAC-learn unambiguous NTS languages, (Clark, 2006)

Membership queries

An efficient query-learning result (Clark, 2010)
Pick a finite set of contexts F
Test if $C_L(u) \cap F = C_L(v) \cap F$

Algorithm

Maintain two sets:

- A set of strings K; includes Σ and λ
- A set of contexts F; includes (λ, λ)

- We have rows for K
- And also for KK – every pair.

Observation table

K a set of strings and F a set of contexts

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										
k_7										
k_6										
k_5										
k_4										
k_3										
k_2										
k_1										

Observation table

K a set of strings and F a set of contexts

	(λ, λ)	(a, λ)	(λ, b)
λ	■		
a			■
b		■	
ab	■		

Observation table

K a set of strings and F a set of contexts

	(λ, λ)	$(aaabb, bccc)$	$(aaabbc, \lambda)$	$(\lambda, abbccc)$	$(aaab, bccc)$	(aa, bbc)	$(aa, bbbc)$	(abb, cc)	$(abbb, cc)$
bcc									■
aab							■		
$bbcc$	■							■	
bc	■							■	
abc	■								
$aabb$	■					■			
ab	■					■			
c	■		■						■
b					■				
a	■			■			■		
λ	■	■				■		■	

Observation table

K a set of strings and *F* a set of contexts

	(aa, bbc)	(λ, λ)	(abb, cc)	$(aaabb, bccc)$	$(aaabbc, \lambda)$	$(\lambda, abbccc)$	$(aaab, bccc)$	$(aa, bbbc)$	$(abbb, cc)$
bcc									■
aab								■	
abc		■							
$aabb$	■	■							
ab	■	■							
λ	■	■	■	■					
$bbcc$		■	■						
bc		■	■						
c		■			■				■
b							■		
a		■				■		■	

Substitutable

$L = \{a^n c b^n | n \geq 0\}$

	(λ,λ)	(a,b)	(λ,cb)	(ac,b)	(a,λ)	(λ,b)	(ac,λ)	(aac,b)
$aacb$						■		
ac						■		
$acbb$					■			
cb					■			
$aacbb$	■	■						
acb	■	■						
c	■	■						
b							■	■
a			■					

Example

Artificial

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Example

Artificial

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Example

Artificial

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Example

Artificial

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Example

Artificial

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Example

Artificial

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										×
k_7										×
k_6										×
k_5				×	×	×				
k_4				×	×	×	×	×	×	
k_3						×	×	×		
k_2	×	×	×							
k_1	×									

Constructing CFG

Given the observation table:

- Non-terminals are equivalence classes of rows: $w \in N$ if N is $[w]$
- Add $N \rightarrow PQ$ if there is a $u \in P, v \in Q$ and $uv \in N$.
- Initial non-terminals are those rows with (λ, λ): $S \rightarrow N$ iff $N \subseteq L$

Example

Dyck language

	(λ, λ)	(a, λ)	(λ, b)
λ	1	0	0
a	0	0	1
b	0	1	0
ab	1	0	0
aab	0	0	1
abb	0	1	0
aa	0	0	0
ba	0	0	0
bb	0	0	0
bab	0	1	0
aba	0	0	1
$abab$	1	0	0

Example

Dyck language

	(λ, λ)	(a, λ)	(λ, b)
λ	1	0	0
ab	1	0	0
$abab$	1	0	0
a	0	0	1
aab	0	0	1
aba	0	0	1
b	0	1	0
abb	0	1	0
bab	0	1	0
aa	0	0	0
ba	0	0	0
bb	0	0	0

Discard rows with no element in K.

Example

Dyck language

Three classes

- $\{\lambda, ab, abab\}$ – Call this S
- $\{a, aab, aba\}$ – A
- $\{b, bab, abb\}$ – B.

Add rules

- $S \rightarrow AB$, $S \rightarrow SS$
- $A \rightarrow AS, SA, a$
- $B \rightarrow BS, SB, b$

Note that there is no rule $X \rightarrow AA$.

Result

Clark, ICGI 2010

Theorem

This algorithm polynomially learns the class of congruential context free languages from MQs and EQs

Language class

A CFG is congruential if $N \stackrel{*}{\Rightarrow} u, N \stackrel{*}{\Rightarrow} v$ implies $u \equiv_L v$

Observation

The same approach works for positive data and membership queries alone

NTS Languages

Boasson and Senizergues

Up to now we have relied on undecidable language theoretic properties. NTS is a decidable syntactic property.

Definition

A grammar G is non terminally separated (NTS) iff for every N, M if $N \stackrel{*}{\Rightarrow} \alpha$ and $M \stackrel{*}{\Rightarrow} u\alpha v$ then $M \stackrel{*}{\Rightarrow} uNv$.

NTS Languages

Boasson and Senizergues

Up to now we have relied on undecidable language theoretic properties. NTS is a decidable syntactic property.

Definition

A grammar G is non terminally separated (NTS) iff for every N, M if $N \stackrel{*}{\Rightarrow} \alpha$ and $M \stackrel{*}{\Rightarrow} u\alpha v$ then $M \stackrel{*}{\Rightarrow} uNv$.

For we like sheep have been led astray.

NTS Languages

Boasson and Senizergues

Up to now we have relied on undecidable language theoretic properties. NTS is a decidable syntactic property.

Definition

A grammar G is non terminally separated (NTS) iff for every N, M if $N \overset{*}{\Rightarrow} \alpha$ and $M \overset{*}{\Rightarrow} u\alpha v$ then $M \overset{*}{\Rightarrow} uNv$.

For we like sheep have been led astray.

Informally

If there is a string like “We like sheep” that can be a sentence, whenever you see an occurrence of “We like sheep” it can be an S.

Congruence classes of NTS languages

Congruence classes of non-terminals

Suppose G is an NTS grammar, and N is a non-terminal, then if $N \stackrel{*}{\Rightarrow} u$ and $N \stackrel{*}{\Rightarrow} v$ then $u \equiv_L v$.

Congruence classes of NTS languages

Congruence classes of non-terminals

Suppose G is an NTS grammar, and N is a non-terminal, then if $N \stackrel{*}{\Rightarrow} u$ and $N \stackrel{*}{\Rightarrow} v$ then $u \equiv_L v$.

- Thus we have a partial equivalence between the congruence classes of the language, and the non-terminals.
- Decidable property of CFG.
- Decidable equivalence

The MAT algorithm can learn all NTS languages.

Unambiguous NTS languages

If we have an *unambiguous* NTS language then:

- Any two strings generated from the same non-terminal will have the same probabilistic distribution.

Ambiguous NTS languages

Some strings in $\{w|N \stackrel{*}{\Rightarrow} w\}$ could have different sets of derivations.
Add production $N \rightarrow w$ with large parameter.

PAC-learning Unambiguous NTS grammars

- If we restrict the distributions to those generated by a PCFG with the same structure then we can learn.

PAC-learning Unambiguous NTS grammars

- If we restrict the distributions to those generated by a PCFG with the same structure then we can learn.
- Parameters
  - μ_1-Distinguishability: same as with PDFAs

PAC-learning Unambiguous NTS grammars

- If we restrict the distributions to those generated by a PCFG with the same structure then we can learn.
- Parameters
  - μ_1-Distinguishability: same as with PDFAs
  - Separability: contexts are sufficiently far apart: A PCFG is ν-separable for some $\nu > 0$ if for every pair of strings u, v in $Sub(L(G))$ such that $u \neq v$, it is the case that $L_\infty(C_u - C_v) \geq \nu \min(L_\infty(C_u), L_\infty(C_v))$

PAC-learning Unambiguous NTS grammars

- If we restrict the distributions to those generated by a PCFG with the same structure then we can learn.
- Parameters
  - μ_1-Distinguishability: same as with PDFAs
  - Separability: contexts are sufficiently far apart: A PCFG is ν-separable for some $\nu > 0$ if for every pair of strings u, v in $Sub(L(G))$ such that $u \not\equiv v$, it is the case that $L_\infty(C_u - C_v) \geq \nu \min(L_\infty(C_u), L_\infty(C_v))$
  - Reachability: A PCFG is μ_2-reachable, if for every non-terminal $N \in V$ there is a string u such that $N \stackrel{*}{\Rightarrow}_G u$ and $L_\infty(C_u) > \mu_2$.

Result

Theorem

The class of unambiguous NTS grammars is PAC-learnable, with parameters μ_1, ν, μ_2, when the distributions are generated by a PCFG with the same structure as the NTS grammar.

Result

Theorem

The class of unambiguous NTS grammars is PAC-learnable, with parameters μ_1, ν, μ_2, when the distributions are generated by a PCFG with the same structure as the NTS grammar.

- Requirement for unambiguity is very strong with NTS grammars.
- Too many stratificational parameters?

Limitations of NTS Grammars

- NTS CFGs are significantly more powerful than PDFAs, and they achieve a better approximation of natural language syntax.
- Like PDFAs, they are efficiently learnable because their primitive elements (non-terminals) and their operations (production rules) are easily inferred from observable linguistic data.
- However, NTS CFGs cannot accommodate mildly context sensitive syntactic properties, which are attested in natural language (Shieber (1985)).
- They also require an excessively large number of syntactic categories, which are too narrow and fine grained in their substring coverage.

Language class

Still limited

Includes

- All regular languages (syntactic monoid is finite)
- Dyck language
- $\{a^n b^n | n \geq 0\} \ldots$

Many simple languages are not in this class:

- Palindromes over $\{a, b\}$
- $L = \{a^n b^n | n \geq 0\} \cup \{a^n b^{2n} | n \geq 0\}$
- $L = \{a^n b^n c^m | n, m \geq 0\} \cup \{a^m b^n c^n | n, m \geq 0\}$

(Some subtle differences in classes but we conjecture that they define the same class)

Linguistic modelling

This seems close to the base model that is used in a lot of empirical work.

Limitations

Inadequate for natural language if Σ is a set of words:

- Exact substitutability is too strict
- Scholz and Pullum (2007): “fond” versus “proud”
- “cat” is not perfectly substitutable with “dog”
- Words are almost never perfectly substitutable; phrases sometimes.

The classes are

- far too small
- they are treated as completely unrelated

$$NP \rightarrow DT\ N$$

	company	companies	associate	furniture	sheep	oil
the	1	1	1	1	1	1
a	1				1	
an			1			1
this	1		1	1	1	1
those		1			1	
some		1		1	1	1

$$NP \rightarrow DT\ N$$

	company	companies	associate	furniture	sheep	oil
the	1	1	1	1	1	1
a	1				1	
an			1			1
this	1		1	1	1	1
those		1			1	
some		1		1	1	1

- Every determiner and noun is in a different congruence class
- Different rule for each combination

Exact congruence

- 'cat' and 'dog' are not exactly congruent:
  $C_L(cat) \neq C_L(dog)$
- but they are very similar: $C_L(cat)$ and $C_L(dog)$ have a lot of elements in common
- We need to look at $C_L(cat) \cap C_L(dog)$.

Outline

Limitations of congruential approach

- Not context sensitive
- Classes are too small: need many non-terminals
- Fails to capture generalisations
- No notion of feature
- Hard to tell very similar classes apart
- Lattice based approaches (Clark FG 2009, CoNLL 2010, ICGI 2010)

Example

“cat” in the sentence “There is a cat over there”

Smallest class

Set of all strings that can be substituted for “cat” in all contexts
Possibly only “cat”

Largest class

The set of all strings that can occur in a single context
“There is a _ over there”

- cat
- dog
- large cat
- large cat near here and a small dog

Multiple contexts

Better

The set of all strings that can occur in **both** contexts

1. “There is a _ over there”
2. “The _ just ran out”

Multiple contexts

Better

The set of all strings that can occur in **both** contexts

1. “There is a _ over there”
2. “The _ just ran out”

Contexts are often “ambiguous”

“He is _”

- boring
- a lawyer
- away on holiday

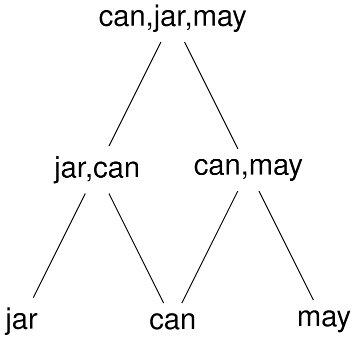
Ambiguity in congruential models

Examples

- Can I have a can of beans?
- May I have a jar of beans?

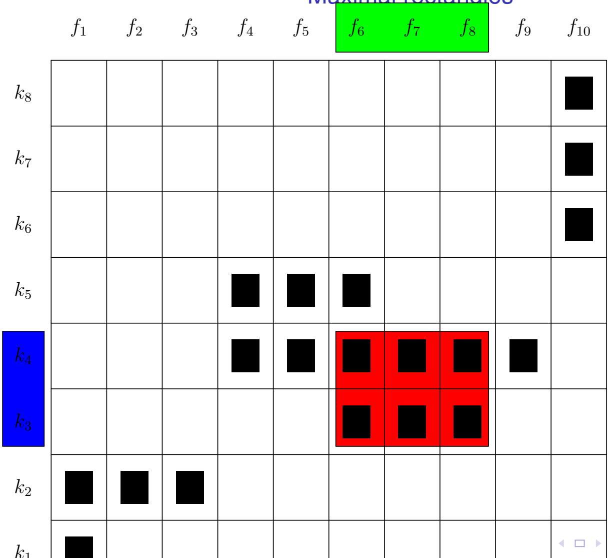
- “can” and “may” are different distributionally
- “can” and “jar” are different distributionally
- The structural descriptions are thus completely distinct

Ambiguity in lattice models



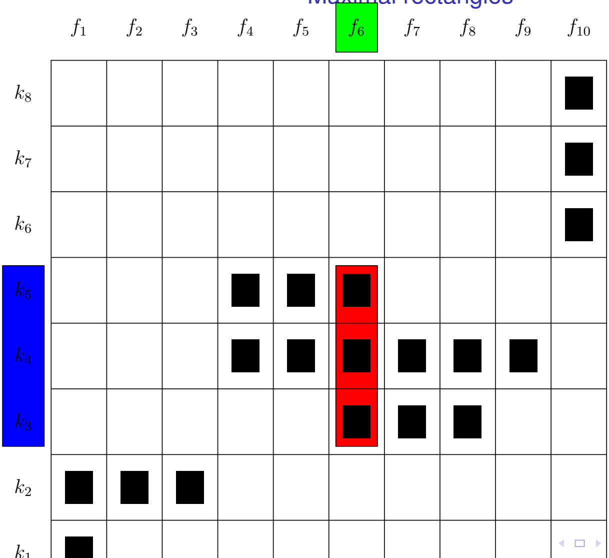
Concepts

Maximal rectangles



Concepts

Maximal rectangles



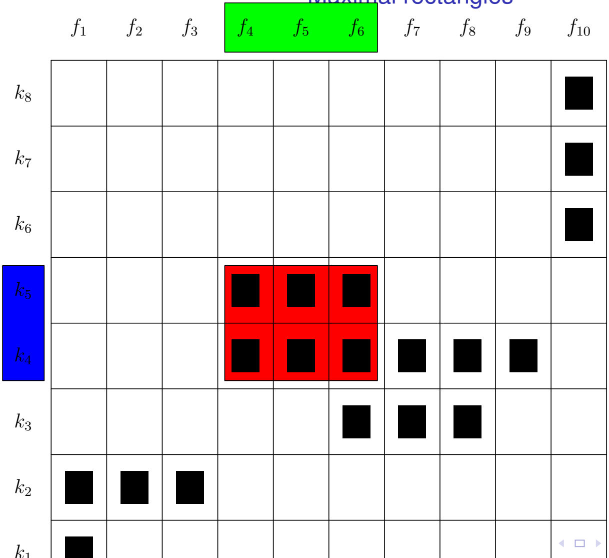
Concepts

Maximal rectangles

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

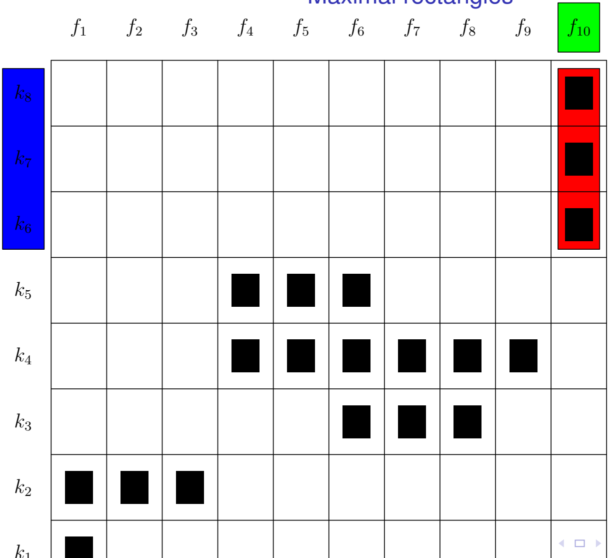
Concepts

Maximal rectangles



Concepts

Maximal rectangles



Concepts

Maximal rectangles

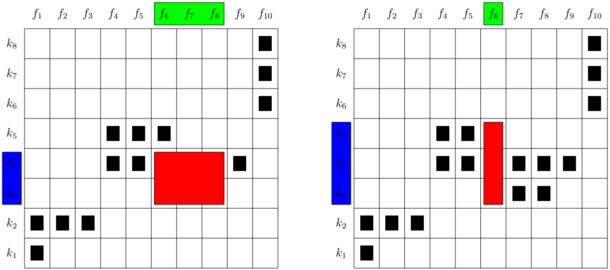
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Concepts

Maximal rectangles

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Partial order



Top and bottom

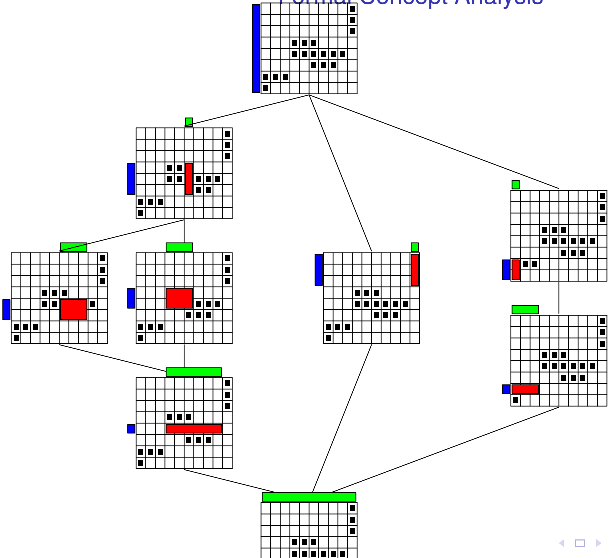
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Top and bottom

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Complete Lattice

Formal Concept Analysis



Rulon Wells

Immediate Constituents, Language 1947

It is easy to define a focus-class embracing a large variety of sequence classes but characterized by only a few environments; it is also easy to define one characterized by a great many environments in which all its members occur but on the other hand poor in the number of diverse sequence-classes that it embraces. What is difficult, but far more important than either of the easy tasks, is to define focus-classes rich both in the number of environments chracterizing them and at the same time in the diversity of sequence classes that they embrace.

- Concepts high up in the lattice have a few contexts, but lots of strings
- Concepts low down have a larger number of contexts, but only a few strings.

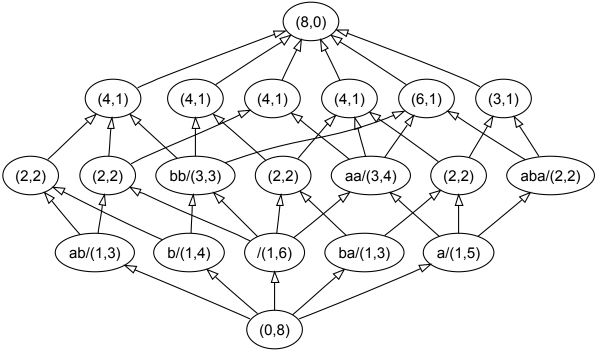
Lattice

Palindrome language over *a*, *b*

	(λ,λ)	(a,λ)	(ab,λ)	(aa,λ)	(b,λ)	(ba,λ)	(λ,b)	(λ,a)
aba	■		■					
bb	■				■		■	
ba		■	■				■	
ab					■	■		■
aa	■	■		■				■
b	■				■	■	■	
a	■	■	■	■				■

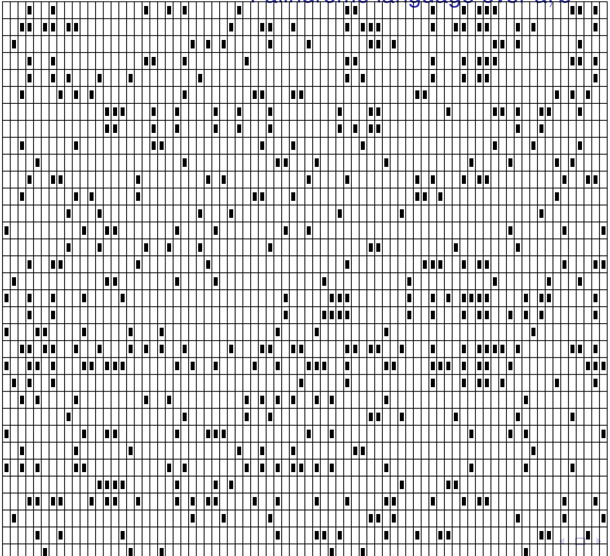
Lattice

Many rectangles

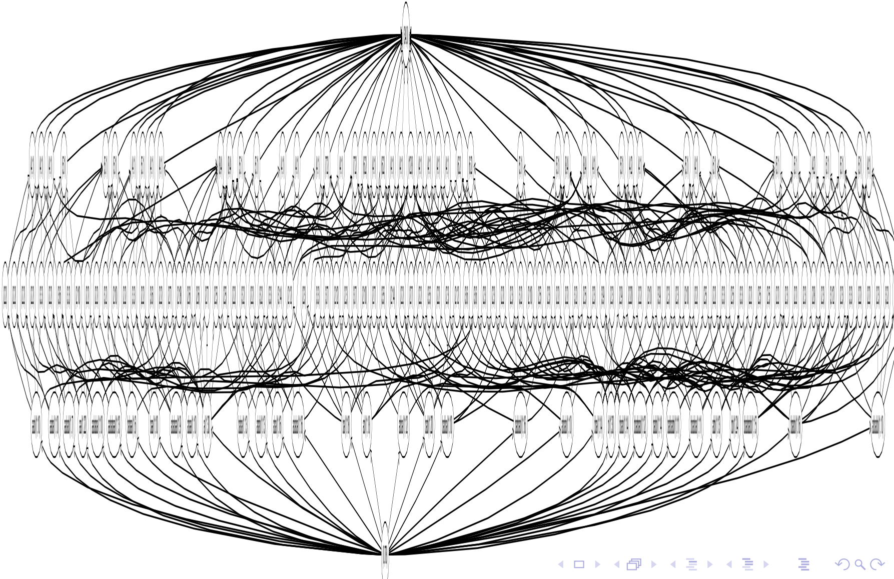


Lattice

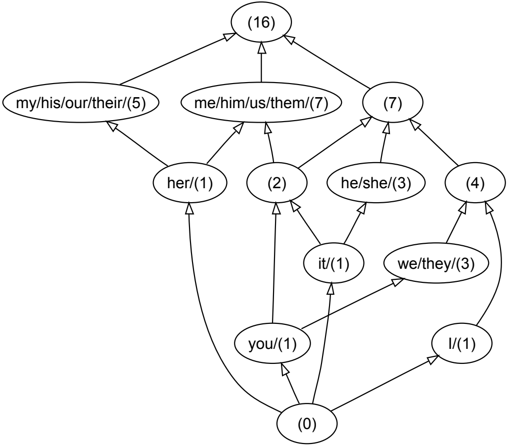
Palindrome language over a, b



Lattice



Linguistic concepts



Non-terminals

On the left hand side

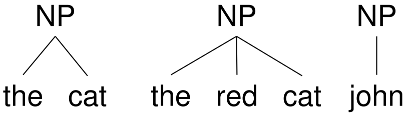
```
NP -> D N
NP -> D Adj N
NP -> N_proper
```

On the right hand side

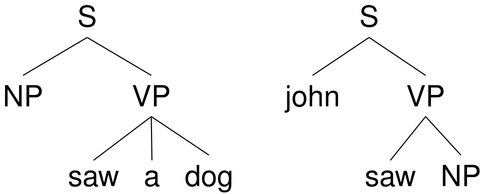
```
S -> NP VP
VP -> V_trans NP
```

Examples

On the left hand side: substrings: $Y(NP)$

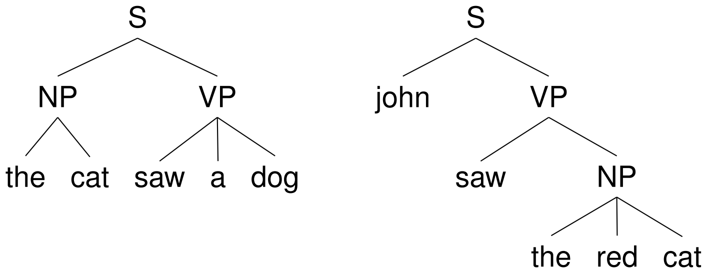


On the right hand side: contexts $C(NP)$



Context free

The term “context-free” means we can combine any context with any substring. Non-terminals are rectangles – almost always concepts.



Context free grammar

Representation

Non-terminals correspond to syntactic concepts.

Context free grammar

Representation

Non-terminals correspond to syntactic concepts.

Rules

$X \rightarrow YZ$

$\langle S_X, C_X \rangle \rightarrow \langle S_Y, C_Y \rangle \langle S_Z, C_Z \rangle$

If and only if $X \supseteq YZ$

Distributional Lattice Grammars

- A problem is that the lattice can be exponentially large.
- However if we shift to a slightly context sensitive formalism we can still compute efficiently.
- We use DLGS: these compute an approximation to the distribution.
- Cubic time parsing — not quite all CFLs.
- Some non context free languages.

Learnability results

Clark, CoNLL 2010

DLGs can be efficiently learned using MQs.

Clark, ICGI 2010

CFGs based on at most f contexts can be polynomially learned with MQs

Yoshinaka, DLT 2011

2 more algorithms for learning CFGs based on the lattice primal/dual variants

Three relations

Regular

$l \sim r$ iff $lr \in L$

Context-substring

$(l, r) \sim u$ iff $lur \in L$

Three relations

Regular

$l \sim r$ iff $lr \in L$

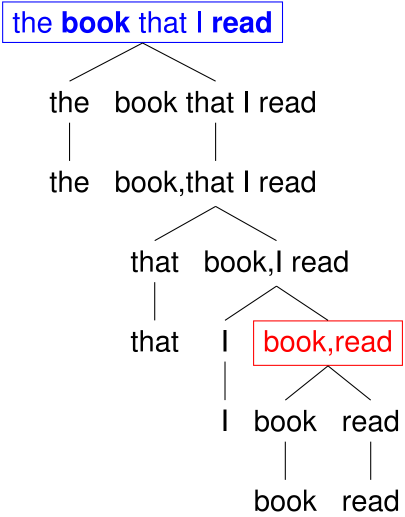
Context-substring

$(l, r) \sim u$ iff $lur \in L$

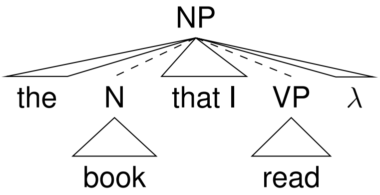
Natural generalisation

$(l, m, r) \sim (u, v)$ iff $lumvr \in L$

Extended contexts



Extended contexts



Multiple context free grammars

Yoshinaka, 2009,2010

“this is the book that I told you to read”

- “this is _ that I told you to _ ”
- (the book, read)

Equivalence of MCFGs and Minimalist Grammars

MGs are weakly and strongly equivalent to MCFGs
Derivation trees of MGs can be learned.
Derived trees can be deterministically generated from the derivation trees
This gives a natural treatment of “movement”

Example

$L = \{cwcw | w \in (a, b)^*\}$

$\langle c, c\rangle \equiv_L \langle ca, ca\rangle \equiv_L \langle cbb, cbb\rangle$

Rules

Combine $\langle c, c\rangle$ with $\langle a, a\rangle$ to get $\langle ca, ca\rangle$

Converts $\langle cw, cw\rangle$ to $cwcw$

Result

Yoshinaka and Clark (2010)

Congruential MCFGs

A congruential MCFG is one where all tuples generated by a non-terminal are congruent.

Result

Congruential MCFGs are polynomially learnable from MQs and EQs.

(way too simple for natural languages, but includes cross-serial dependencies, MIX languages)
Needs to be combined with the lattice approach.

Further results

These results can be generalised to learning other types of problem:

Yoshinaka and Kanazawa, LACL 2011

3 results for abstract categorial grammars

Yoshinaka and Kasprzik, DLT 2011

Learning context free tree languages

Clark ICML 2011

Learning context free transducers from input output pairs

Outline

Weak learning

Two reasonable conceptions

Weak learning of strings

Tractable but ignores the role of semantics

Weak learning of sound/meaning pairs

Tractable (Yoshinaka and Kanazawa, 2011)
Assumes that learners have complete access to meanings

- Implausible even for adults
- Language acquisition starts before children plausibly have the cognitive resources to do the relevant inferences.

We observe convergence in these senses

Strong learning I

Strong learning

Inputs are strings, output generates "correct structures"
Intractable but irrelevant

We do not observe strong learning

- Different speakers who have converged to identical sets of sound/meaning pairs might assign slightly different structures.
- For example, memorize different chunks

Psycholinguistic data

Experimental data

Other than from the set of sound/meaning pairs

- Click Perception (Fodor and Bever, 1965)
- Structural Priming (Bock, 1986)
- Neuroimaging (Tettamanti et al. 2002)

Evidence for some types of hierarchically structured representations but don't provide evidence on inter-speaker convergence.

Strong learning II

But we do need to have some structural descriptions:

- Hierarchically structured representations that support semantic interpretation
  - Ambiguity
  - Displaced constituents – “movement”
- Learnable with limited or no information about the semantics
- Unrealistic to expect to be able to distinguish spurious ambiguity from semantic ambiguity
- Don’t require exact convergence

Inputs for weak learning

Assumption

We have some well defined set of strings of phonemes.
$L \subseteq \Sigma^*$
Assume:

- L is the set of syntactically well formed utterances
- Semantic constraints are handled by another component

Inputs revisited

- My aunt is pregnant
- My toothbrush is covered with toothpaste

Inputs revisited

- My aunt is pregnant
- My toothbrush is covered with toothpaste

- # My aunt is covered with toothpaste
- # My toothbrush is pregnant

The input to the child consists largely of semantically well formed utterances.

English example

- Mary kicked the ball and Jill ate the cake

English example

- Mary kicked the ball and Jill ate the cake
- # Mary fed the ball and Jill ate the cake
- # Mary kicked the cake and Jill ate the cake
- # Mary fed the ball and Jill ate the cat
- Mary fed the cat and Jill ate the cake

Semantic

Proposition

The right dependencies can be inferred if the child gets semantically well formed utterances

- Weak learning of the set of semantically well formed utterances gives you strong learning implicitly.
- But how can we make these dependencies explicit?

Structural descriptions from Distributional Learning

Congruential approach

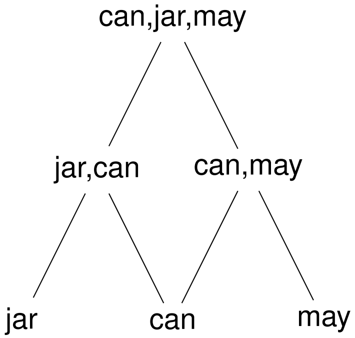
Equivalence classes of strings that are distributionally identical

Lattice based approach

Sets of strings that have some shared distribution
A hierarchy of sets of strings:

- Large sets of strings that have a small set of contexts in common
- Smaller sets of strings that are very similar or identical

Ambiguity in lattice models



Lattice labeled trees

Structural descriptions that are ordered trees:

- Each node is labeled with a concept (set of strings) that contains the yield.
- The concept of each node must *properly* contain the concatenation of the concepts of the children of that node.
- The root node must be the concept that consist of the language.
- Now we have a choice of labels within a fixed tree.
- **We want labels to be maximal subject to the above constraints.**

Classic Example

- John is eager to please
- John is easy to please

Classic Example

- John is eager to please
- John is easy to please

- John is ready to eat
- the soup is ready to eat
- the chicken is ready to eat

Classic Example

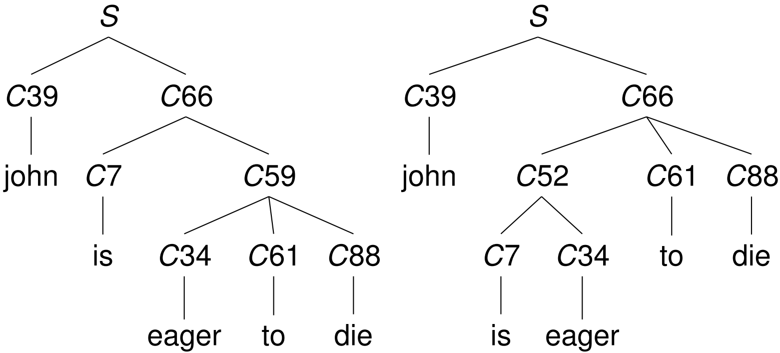
- John is eager to please
- John is easy to please

- John is ready to eat
- the soup is ready to eat
- the chicken is ready to eat

Methodology Toy grammar; learn CFG from examples; derived set of all valid trees.

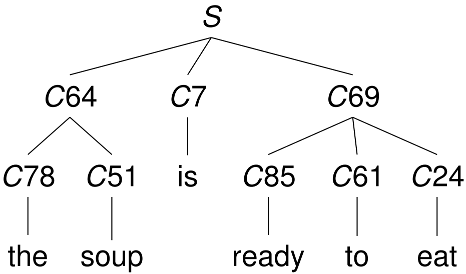
Trees

John is eager to die



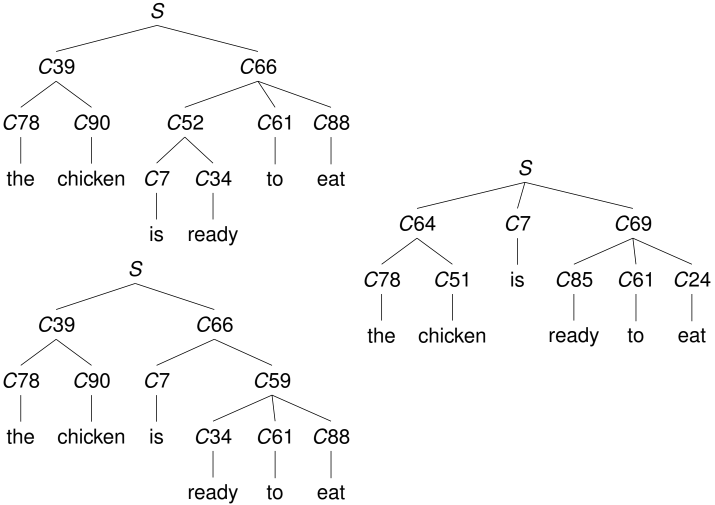
Trees

the soup is ready to eat



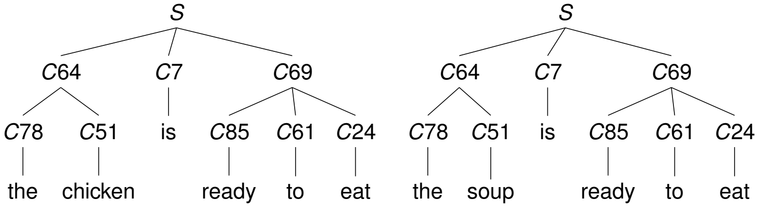
Trees

the chicken is ready to eat



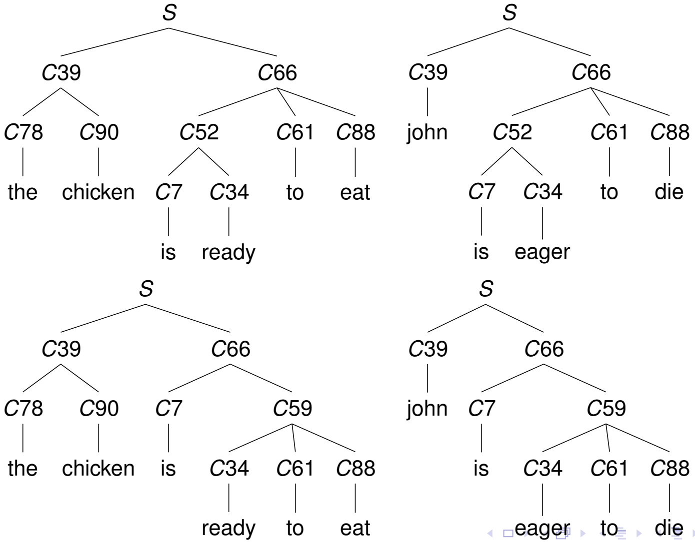
Trees

Comparison 1



Trees

Comparison 2



Outline

Developing Efficiently Learnable Representations

Slogan

The structure of the representation should be based on the observable structure of the language itself.

- Representations whose rules and primitives are based on properties of the data can be easily inferred from data.
- Much of the work that we describe here follows a line of inquiry that has its origin in Angluin (1982)'s pioneering research on the efficient learning of reversible regular languages.