Active Learning

Colin de la Higuera

2010
Outline

1. Motivations and applications
2. The learning model
3. Some negative results
4. Algorithm L*
5. Some implementation issues
6. Extensions
7. Conclusion
0 General idea

- The learning algorithm (he) is allowed to interact with his environment through queries
- The environment is formalised by an oracle (she)
- Also called learning from queries or oracle learning
1 Motivations
Goals

- define a credible learning model
- make use of additional information that can be measured
- explain thus the difficulty of learning certain classes
- solve real life problems
Application: robotics

- A robot has to find a route in a maze
- The maze is represented as a graph
- The robot finds his way and can experiment
- The robot gets feedback

Application: web wrapper induction

- System SQUIRREL learns tree automata
- Goal is to learn a tree automaton which, when run on XML, returns selected nodes

Applications: under resourced languages

- When a language does not have enough data for statistical methods to be of interest, use a human expert for labelling.
- This is the case for many languages.
- Examples
  - Interactive predictive parsing
  - Computer aided translation
Model Checking

- An electronic system can be modelled by a finite graph (a DFA)
- Checking if a chip meets its specification can be done by testing or by trying to learn the specification with queries

Playing games


2. The model
Running example

- Suppose we are learning DFA
- The *running example* for our target is:
The Oracle

- knows the language and has to answer correctly
- no probabilities unless stated
- worst case policy: the Oracle does not want to help
Some queries

1. sampling queries
2. presentation queries
3. membership queries
4. equivalence queries (weak or strong)
5. inclusion queries
6. correction queries
7. specific sampling queries
8. translation queries
9. probability queries
2.1 Sampling queries (Ex)

\[ w \sim l_T(w) \]

\( w \) is drawn following some unknown distribution
Sampling queries (Pos)

$x$ is drawn following some unknown distribution, restricted to $L(T)$
Sampling queries (Neg)

$X$ is drawn following some unknown distribution, restricted to $\Sigma^* \setminus L(T)$
Example

- Ex() might return $(aabab, 1)$ or $(\lambda, 0)$
- Pos() might return $abab$
- Neg() might return $aa$

Requires a distribution over $\Sigma^*$, $L(T)$ or $\Sigma^* \setminus L(T)$
2.2 Presentation queries

- A presentation of a language is an enumeration of all the strings in $\Sigma^*$, with a label indicating if a string belongs or not to $L(T)$ (informed presentation),
- or an enumeration of all the strings in $L(T)$ (text presentation)
- There can be repetitions
Presentation queries

\[ i \in \mathbb{N} \quad \text{\textbf{\textit{w}}\textbf{\textit{\right}} f(i)} \]

*f is a valid (unknown) presentation.*

Sub-cases can be text or informed presentations.
Example

- \( \text{Pres}_{\text{text}}(3) \) could be \( bba \)
- \( \text{Pres}_{\text{text}}(17) \) could be \( abbaba \)
- (the « selected » presentation being \( b, ab, aab, bba, aaab, abab, abba, bbab, \ldots\) )
Example

- $\text{Pres}_{\text{informed}}(3)$ could be $(aab,1)$
- $\text{Pres}_{\text{informed}}(1)$ could be $(a,0)$
- (the « selected » presentation being $(b,1),(a,0),(aaa,0),(aab,1),(bba,1),(a,0)\ldots$)
2.3 Membership queries.

\[ x \in L(T) \]

Yes if \( x \in L(T) \)

No if not \( L(T) \) is the target language
Example

- \( \text{MQ}(aab) \) returns 1 (or true)
- \( \text{MQ}(bbb) \) returns 0 (or false)
2.4 Equivalence (weak) queries.

Yes if $L(H) = L(T)$
No if $\exists x \in \Sigma^*: x \in L(T) \oplus L(H)$

$A \oplus B$ is the symmetric difference
Equivalence (strong) queries.

Yes if \( T \equiv H \)

\[ x \in \Sigma^*: \quad x \in L(H) \oplus L(T) \quad \text{if not} \]
Example

- EQ($H$) returns $abbb$ (or $abba...$)
- WEQ($H$) returns false
2.5 Subset queries.

Yes if \( L(H) \subseteq L(T) \)

\( x \in \Sigma^* : x \in L(H) \land x \not\in L(T) \)

if not
Example

- $SSQ(H_1)$ returns true
- $SSQ(H_2)$ returns $abbb$
2.6 Correction queries.

$x \in \Sigma^*$

Yes if $x \in L(T)$

$y \in L(T)$: $y$ is a correction of $x$ if not


Example

- $CQ_{suff}(bb)$ returns $bba$
- $CQ_{edit}(bb)$ returns any string in $\{b,ab,bba\}$
- $CQ_{edit}(bba)$ and $CQ_{suff}(bba)$ return true
2.7 Specific sampling queries

- Submit a grammar $G$
- Oracle draws a string from $L(G)$ and labels it according to $T$
- Requires an unknown distribution

- Allows for example to sample string starting with some specific prefix
2.8 Probability queries

- Target is a PFA.
- Submit \( w \), Oracle returns \( \Pr \mathcal{A}(w) \)

String \( ba \) should have a relative frequency of \( 1/16 \)
2.9 Translation queries

- Target is a transducer
- Submit a string. Oracle returns its translation

\[
\begin{align*}
    \text{Tr}(ab) & \text{ returns } 100 \\
    \text{Tr}(bb) & \text{ returns } 0001
\end{align*}
\]
Learning setting

Two things have to be decided

- The exact queries the learner is allowed to use
- The conditions to be met to say that learning has been achieved
What queries are we allowed?

- The combination of queries is declared
- Examples:
  - $Q=\{MQ\}$
  - $Q=\{MQ,EQ\}$ (this is an MAT)
Defining learnability

- Can be in terms of classes of languages or in terms of classes of grammars
- The size of a language is the size of the smallest grammar for that language
- Important issue: when does the learner stop asking questions?
You can’t learn DFA with membership queries

- Indeed, suppose the target is a finite language
- Membership queries just add strings to the language
- But you can’t stop and be sure of success
Correct learning

A class $\mathcal{C}$ is learnable with queries from $\mathcal{Q}$ if there exists an algorithm $\alpha$ such that:

$\forall L \in \mathcal{C}, \alpha$ makes a finite number of queries from $\mathcal{Q}$, halts and returns a grammar $G$ such that $L(G) = L$

We say that $\alpha$ learns $\mathcal{C}$ with queries from $\mathcal{Q}$
Polynomial update

\[ \langle p(|T|) \rangle \]

\[ \langle p(\max(|x_1|, |x_2|, |x_3|, |x_4|)) \rangle \]
Correct learning

A class $\mathcal{C}$ is learnable with a polynomial number of queries from $\mathcal{Q}$ if there exists an algorithm $a$ and a polynomial $q(\cdot, \cdot)$ such that:

1) $\forall L \in \mathcal{C} \ a$ learns $L$ with queries from $\mathcal{Q}$

2) $a$ makes polynomial updates

3) $a$ uses during a run $\rho$ at most $q(m, |L|)$ queries, where $m = mInfo(\rho)$
Comment

- In the previous definitions, the queries receive deterministic answers.
- When the queries have a probabilistic nature, things still have to be discussed as identification cannot be sure.
3 Negative results
3.1 Learning from membership queries alone

- Actually we can use subset queries and weak equivalence queries also, without doing much better.
- Intuition: keep in mind lock automata...
Lemma (Angluin 88)

- If a class $\mathcal{C}$ contains a set $\mathcal{C} \cap$ and $n$ sets $\mathcal{C}_1...\mathcal{C}_n$ such that $\forall \ i, j \in [n] \quad \mathcal{C}_i \cap \mathcal{C}_j = \mathcal{C} \cap$, any algorithm using membership, weak equivalence and subset queries needs in the worse case to make $n-1$ queries.
WEQ(\(C \cap \))
WEQ($C_j$)

Equivalence query on this $C_j$
Is $C \cap \subseteq$ included?

**YES!!!**

*(so what?)*
Subset query on this $C_j$
Does $x$ belong?

YES!!!

(so what?)

$MQ(x)$
Does \( x \) belong?

No... of course.

\( MQ(x) \)
3.2 What about equivalence queries?

- *Negative results for Equivalence Queries, D. Angluin, Machine Learning, 5, 121-150, 1990*

- Equivalence queries measure also the number of implicit prediction errors a learning algorithm might make
Proof (summarised)

<table>
<thead>
<tr>
<th>Query</th>
<th>Answer</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WEQ(C_i)$</td>
<td>No</td>
<td>eliminates $C_i$</td>
</tr>
<tr>
<td>$SSQ(C \cap)$</td>
<td>Yes</td>
<td>eliminates nothing</td>
</tr>
<tr>
<td>$SSQ(C_i)$</td>
<td>No</td>
<td>eliminates nothing</td>
</tr>
<tr>
<td>$MQ(x) (\in C \cap)$</td>
<td>Yes</td>
<td>eliminates nothing</td>
</tr>
<tr>
<td>$MQ(x) (\notin C \cap)$</td>
<td>No</td>
<td>eliminates $C_i$ such that $x \in C_i$</td>
</tr>
</tbody>
</table>
Corollary

Let $DFA_n$ be the class of DFA with at most $n$ states

$DFA_n$ cannot be identified by a polynomial number of membership, weak equivalence and inclusion queries.

- $L \cap = \emptyset$
- $L_i = \{w_i\}$ where $w_i$ is $i$ written in base 2
Algorithm L*

Learning regular sets from queries and counter-examples, D. Angluin, Information and computation, 75, 87-106, 1987
Queries and Concept learning, D. Angluin, Machine Learning, 2, 319-342, 1988
4.1 The Minimal Adequate Teacher

- Learner is allowed:
  - strong equivalence queries
  - membership queries
General idea of $L^*$

- find a good table (representing a DFA)
- submit it as an equivalence query
- use counterexample to update the table
- submit membership queries to make the table good
- iterate
### 4.2 An observation table

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$aa$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ab$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
The states (RED)

<table>
<thead>
<tr>
<th></th>
<th>λ</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>aa</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The transitions (BLUE)

The experiments (EXP)
Meaning

<table>
<thead>
<tr>
<th></th>
<th>( \lambda )</th>
<th>( a )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>( a )</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>( b )</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>( aa )</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>( ab )</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>

\[ \delta(q_0, \lambda.\lambda) \in F \Leftrightarrow \lambda \in \mathcal{L} \]

Must have identical label (redundancy)
\[ \delta(q_0, ab.a) \not\in F \iff aba \not\in L \]
Equivalent prefixes

<table>
<thead>
<tr>
<th></th>
<th>(\lambda)</th>
<th>(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda)</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>(aa)</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>(ab)</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>

These two rows are equal, hence

\[ \delta(q_0, \lambda) = \delta(q_0, ab) \]
Equivalent prefixes are states

<table>
<thead>
<tr>
<th></th>
<th>λ</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>1 0</td>
<td>0 0</td>
</tr>
<tr>
<td>a</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>aa</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>ab</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>
Building a DFA from a table

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>$a$</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>$aa$</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>$ab$</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>
$\begin{array}{c|cc}
\lambda & a & b \\
\hline
\lambda & 1 & 0 \\
a & 0 & 0 \\
b & 1 & 0 \\
aa & 0 & 0 \\
ab & 1 & 0 \\
\end{array}$
Some rules

<table>
<thead>
<tr>
<th></th>
<th>λ</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>aa</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

This set is suffix-closed

RED \RED = BLUE
## An incomplete table

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$aa$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$ab$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Good idea

We can complete the table by submitting membership queries...

\[
\begin{array}{c|c}
u & \checkmark \\
\hline
? & \end{array}
\]

Membership query:

\[uv \in L \ ?\]
A table is **closed** if any row of **BLUE** corresponds to some row in **RED**

<table>
<thead>
<tr>
<th></th>
<th>(\lambda)</th>
<th>(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(a)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(b)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(aa)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(ab)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
And a table that is not *closed*.

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>$a$</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>$aa$</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>$ab$</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram](image)
What do we do when we have a table that is not closed?

- Let $s$ be the row (of BLUE) that does not appear in RED.
- Add $s$ to RED, and $\forall a \in \Sigma$, add $sa$ to BLUE.
An inconsistent table

<table>
<thead>
<tr>
<th></th>
<th>( \lambda )</th>
<th>( a )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>( a )</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>( b )</td>
<td>0 0</td>
<td></td>
</tr>
</tbody>
</table>

Are \( a \) and \( b \) equivalent?
A table is consistent if

Every equivalent pair of rows in \( RED \) remains equivalent in \( RED \cup BLUE \) after appending any symbol

\[
OT[s_1] = OT[s_2] \\
\Rightarrow \\
\forall a \in \Sigma, \ OT[s_1a] = OT[s_2a]
\]
What do we do when we have an inconsistent table?

Let $a \in \Sigma$ be such that $OT[s_1] = \text{row}(s_2)$ but $OT[s_1a] \neq OT[s_2a]$

- If $OT[s_1a] \neq OT[s_2a]$, it is so for experiment $e$
- Then add experiment $ae$ to the table
What do we do when we have a closed and consistent table?

- We build the corresponding DFA
- We make an equivalence query!!!
What do we do if we get a counter-example?

- Let \( u \) be this counter-example

- \( \forall w \in \text{Pref}(u) \) do
  - add \( w \) to \( \text{RED} \)
  - \( \forall a \in \Sigma, \text{such that } wa \notin \text{RED} \) add \( wa \) to \( \text{BLUE} \)
4.3 Run of the algorithm

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
</tr>
</tbody>
</table>

Table is now closed and consistent.
An equivalence query is made!

Counter example \textit{baa} is returned
Not consistent
Because of
<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$ba$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$baa$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$bb$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$bab$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$baaa$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$baab$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table is now closed and consistent.
The algorithm

while not A do
    while OT is not complete, consistent or closed do
        if OT is not complete then make MQ
        if OT is not consistent then add experiment
        if OT is not closed then promote
    A$\leftarrow$EQ(OT)
4.4 Proof of the algorithm
Termination / Correctness

- For every regular language there is a unique minimal DFA that recognizes it.
- Given a closed and consistent table, one can generate a consistent DFA.
- A DFA consistent with a table has at least as many states as different rows in $H$.
- If the algorithm has built a table with $n$ different rows in $H$, then it is the target.
Finiteness

- Each closure failure adds one different row to RED
- Each inconsistency failure adds one experiment, which also creates a new row in RED
- Each counterexample adds one different row to RED
Polynomial

- $|\text{EXP}| \leq n$
- at most $n-1$ equivalence queries
- $|\text{membership queries}| \leq n(n-1)m$ where $m$ is the length of the longest counter-example returned by the oracle
Conclusion

- With an MAT you can learn DFA
  - but also a variety of other classes of grammars
  - it is difficult to see how powerful is really an MAT
  - probably as much as PAC learning
  - Easy to find a class, a set of queries and provide and algorithm that learns with them
  - more difficult for it to be meaningful
- Discussion: why are these queries meaningful?
Discussion

- Are membership and equivalence queries realistic?
- Membership queries are plausible in a number of applications
- Equivalence queries are not
- A way around this is to do sampling
Good idea

- If we sample following $\mathcal{D}$ 100 strings, and we coincide in labelling with the Oracle, then how bad are we?

- Formula: suppose the error is more than $\epsilon$, then coinciding 100 times has probability at least $(1-\epsilon)^{100}$. The chance this happens is less than 0.6% for $\epsilon = 5\%$
Conclusion

- If we can draw according to the true distribution, one can learn a approximately correct DFA from membership queries only.
5 Implementation issues

How to implement the table

About Zulu
Use pointers

<table>
<thead>
<tr>
<th></th>
<th>(\lambda)</th>
<th>(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(b)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(ba)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(baa)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(a)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(bb)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(bab)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(baaa)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(baab)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(\lambda)</th>
<th>(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(b)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(aa)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(ba)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(bb)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(baa)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(bab)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(baaa)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(baab)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(baba)</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Zulu competition

- 23 competing algorithms, 11 players
- End of the competition a week ago
- Tasks:
  - Learn a DFA, be as precise as possible, with \( n \) queries
## Results

<table>
<thead>
<tr>
<th>Task</th>
<th>queries</th>
<th>alphabet</th>
<th>Best%</th>
<th>states</th>
<th>Task</th>
<th>queries</th>
<th>alphabet</th>
<th>states</th>
<th>Best%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>304</td>
<td>3</td>
<td>100,00</td>
<td>8</td>
<td>13</td>
<td>725</td>
<td>15</td>
<td>10</td>
<td>100,00</td>
</tr>
<tr>
<td>2</td>
<td>199</td>
<td>3</td>
<td>100,00</td>
<td>16</td>
<td>14</td>
<td>1365</td>
<td>15</td>
<td>17</td>
<td>100,00</td>
</tr>
<tr>
<td>3</td>
<td>1197</td>
<td>3</td>
<td>96,50</td>
<td>81</td>
<td>15</td>
<td>5266</td>
<td>15</td>
<td>60</td>
<td>100,00</td>
</tr>
<tr>
<td>4</td>
<td>1384</td>
<td>3</td>
<td>93,22</td>
<td>100</td>
<td>16</td>
<td>7570</td>
<td>15</td>
<td>71</td>
<td>100,00</td>
</tr>
<tr>
<td>5</td>
<td>1971</td>
<td>3</td>
<td>85,89</td>
<td>151</td>
<td>17</td>
<td>17034</td>
<td>15</td>
<td>147</td>
<td>100,00</td>
</tr>
<tr>
<td>6</td>
<td>3625</td>
<td>3</td>
<td>100,00</td>
<td>176</td>
<td>18</td>
<td>16914</td>
<td>15</td>
<td>143</td>
<td>87,94</td>
</tr>
<tr>
<td>7</td>
<td>429</td>
<td>5</td>
<td>100,00</td>
<td>15</td>
<td>19</td>
<td>1970</td>
<td>5</td>
<td>93</td>
<td>81,67</td>
</tr>
<tr>
<td>8</td>
<td>375</td>
<td>5</td>
<td>100,00</td>
<td>18</td>
<td>20</td>
<td>1329</td>
<td>5</td>
<td>61</td>
<td>70,00</td>
</tr>
<tr>
<td>9</td>
<td>2524</td>
<td>5</td>
<td>96,44</td>
<td>84</td>
<td>21</td>
<td>571</td>
<td>5</td>
<td>40</td>
<td>69,22</td>
</tr>
<tr>
<td>10</td>
<td>3021</td>
<td>5</td>
<td>100,00</td>
<td>90</td>
<td>22</td>
<td>735</td>
<td>5</td>
<td>57</td>
<td>65,11</td>
</tr>
<tr>
<td>11</td>
<td>5428</td>
<td>5</td>
<td>99,94</td>
<td>153</td>
<td>23</td>
<td>483</td>
<td>5</td>
<td>73</td>
<td>86,61</td>
</tr>
<tr>
<td>12</td>
<td>4616</td>
<td>5</td>
<td>100,00</td>
<td>123</td>
<td>24</td>
<td>632</td>
<td>5</td>
<td>78</td>
<td>100,00</td>
</tr>
</tbody>
</table>
6 Further challenges
Transducer learning

abaab → acbbb

abaab → acbbb
Transducer learning

\[ \lambda \xrightarrow{0::0} 0 \xrightarrow{1::1} 1 \xrightarrow{1::0} 0 \xrightarrow{1::10} 10 \]

Multiplication by 3
Typical queries

- Translation queries
- \( \text{Tr}(w)? \) Oracle answers with the translation of \( w \)

PFA learning

- Probabilistic finite automata can be
  - Deterministic
  - Non deterministic
A deterministic PFA

\[
\text{Pr}_A(abab) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{3} \times \frac{2}{3} \times \frac{3}{4} = \frac{1}{24}
\]
A nondeterministic PFA

\[
\text{Pr}(aba) = 0.7 \times 0.4 \times 0.1 \times 1 + 0.7 \times 0.4 \times 0.45 \times 0.2 = 0.028 + 0.0252 = 0.0532
\]
What queries should we consider?

- Probability queries
  - PQ(w)? Oracle returns Pr_D(w)
- EX()? Oracle returns a string w randomly drawn according to D
- Specific sampling query SSQ(L)
  - Oracle returns a string belonging to L sampled according to D
  - Distribution is Pr_D(w)=Pr_D(L)/Pr_D(L)
Context-free grammar learning

- Typical query corresponds to using the grammar (structural query)

- In which case the goal is to identify the grammar, not the language!
7 General conclusion
Some open problems

- Find better definitions
- Do these definitions give a general framework for grammatical inference?
- How can we use resource bounded queries?
- Use Zulu or develop new tools for Zulu
Acknowledgements

- Laurent Miclet, Jose Oncina and Tim Oates for collaboration previous versions of these slides.
- Rafael Carrasco, Paco Casacuberta, Rémi Eyraud, Philippe Ezequel, Henning Fernau, Thierry Murgue, Franck Thollard, Enrique Vidal, Frédéric Tantini,...
- List is necessarily incomplete. Excuses to those that have been forgotten.

http://pagesperso.lina.univ-nantes.fr/~cdlh/slides/

Book, chapters 9 and 13