Learning $\kappa$-testable languages
Acknowledgements

• Laurent Miclet, Jose Oncina and Tim Oates for 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://eurise.univ-st-etienne.fr/~cdlh/slides
Remember

• Regular languages cannot be identified from positive examples only (Gold 67);
• We have to concentrate on a sub-class of these.
**K-testable languages**

Inference of k-Testable Languages in the Strict Sense and Application to Syntactic Pattern Recognition. García & Vidal et al. 1990

Concept initially introduced for Pattern Recognition tasks
**Definition**

Let $k \geq 0$, a $k$-testable machine in the strict sense (k-TSS) is a 4-tuple $Z_k = (\Sigma, I, F, T)$ with:

- $\Sigma$ a finite alphabet
- $I, F \subseteq \Sigma^<k$ (allowed prefixes of length less than $k$ and suffixes of length $k-1$) and also all strings of length less than $k$.
- $T \subseteq \Sigma^k$ (allowed segments)
• The \( k \)-testable language is

\[
L(Z_k) = \left\{ \Sigma^* \cap \Sigma^* (F \cap \Sigma^{k-1}) - \Sigma^* (\Sigma^k \cap T) \Sigma^* \right\} \cup F
\]

• Strings (of length at least \( k \)) have to use all good prefixes and a good suffix of length \( k-1 \), and all sub-strings have to belong to \( T \). Strings of length less than \( k \) should be in \( F \).
• Or: \( \Sigma^k - T \) defines the prohibited segments.

• Key idea: use a window of size \( k \).
An example

This is a 2 testable language. You need a window of size 2 to avoid substring $bb$.

$I = \{ \lambda, a \}$

$F = \{ a \}$

$T = \{ aa, ab, ba \}$
The hierarchy of \( k\text{-TSS} \) languages

- \( k\text{-TSS}(\Sigma) = \{ L \subseteq \Sigma^*: L \text{ is } k\text{-TSS}\} \)
- All finite languages are in \( k\text{-TSS}(\Sigma) \) if \( k \) is large enough!
- \( k\text{-TSS}(\Sigma) \subset [k+1]\text{-TSS}(\Sigma) \)
- \( (ba^k)^* \in [k+1]\text{-TSS}(\Sigma) \)
- \( (ba^k)^* \notin k\text{-TSS}(\Sigma) \)
A language that is not $\kappa$-testable
Given a sample $X$, $a_{k-TSS}(X) = L(Z_k)$ where $Z_k = (\Sigma(X), I(X), F(X), \Pi(X))$ and

- $\Sigma(X)$ is the alphabet used in $X$
- $I(X) = \Sigma(X) <^k \cap \text{Pref}(X)$
- $F(X) = \Sigma(X)^{k-1} \cap \text{Suff}(X) \cup \Sigma(X) <^k \cap X$
- $\Pi(X) = \Sigma(X)^k \cap \{v: uvw \in X\}$
Example

• $X = \{a, \text{aa}, \text{abba}, \text{abbbba}\}$

• Let $k = 3$
  
  - $\Sigma(X) = \{a, b\}$
  
  - $I(X) = \{\lambda, a, \text{aa}, \text{ab}\}$
  
  - $F(X) = \{a, \text{aa}, \text{ba}\}$
  
  - $T(X) = \{\text{abb}, \text{bbb}, \text{bba}\}$

• Hence $a_{k-\text{TSS}}(X) = \text{ab}^*a + a$
Building the corresponding automaton

• Each string in $I$ is a state;
• Each substring of length $k-1$ of strings in $T$ is a state;
• $\lambda$ is the initial state;
• Add a transition labeled $b$ from $u$ to $ub$ for each $ub$ in $I$;
• Add a transition labeled $b$ from $au$ to $ub$ for each $aub$ in $T$;
• Each state/substring that is in $F$ is a final state.
Running the algorithm

\[ X = \{a, \text{aa}, \text{abba}, \text{abbbba}\} \]

\[ I = \{\lambda, a, \text{aa}, \text{ab}\} \]

\[ F = \{a, \text{aa}, \text{ba}\} \]

\[ T = \{a\text{bb}, b\text{bb}, b\text{ba}\} \]
Properties (1)

\[ X \subseteq a_{k-TSS}(X) \]

\( a_{k-TSS}(X) \) is the smallest \( k \)-TSS language that contains \( X \)

- If there is a smaller one, some prefix, suffix or substring has to be absent.
Properties (2)

\( a_{k-TSS} \) identifies any \( k \)-TSS language in the limit.

Once all the prefixes, suffixes and substrings have been seen, the correct automaton is returned.

- If \( Y \subseteq X \), \( a_{k-TSS}(Y) \subseteq a_{k-TSS}(X) \)
Properties (3)

\[ a_{k+1}^{-TSS}(X) \subseteq a_k^{-TSS}(X) \]

- In \( I_{k+1} \) (resp. \( F_{k+1} \) and \( T_{k+1} \)) there are less allowed prefixes (resp. suffixes or substrings) than in \( I_k \) (resp. \( F_k \) and \( T_k \)).

- Notice \( k^{-TSS}(\Sigma) \subset [k+1]^{-TSS}(\Sigma) \)

\[ \forall k > \max_{x \in X} |x|, \quad a_k^{-TSS}(X) = X \]

- Because for a large \( k \), \( T_k(X) = \emptyset \)
Extensions

- These languages have been studied and adapted to:
  - Local languages
  - \( N \)-grams
  - Tree languages
Exercises (1)

- Run $a_{k-TSS}(X)$ for
  - $k=1, 2, 3, \text{ and } 15, \text{ and}$
  - $X=\{ab, \; abab, \; abababab\}$
Exercises (2)

• What is the complexity of $a_{k-TSS}$?
• Give an algorithm that computes a characteristic sample for $a_{k-TSS}$.
• How many mind changes does algorithm $a_{k-TSS}$ make?
Exercises (3)

• How many implicit prediction errors does algorithm $a_{k-TSS}$ make?
• Prove that identifying the entire class of testable languages is impossible from text.
• Prove that learning the entire class, from an informant, can be done but is not polynomial.