LEARNING FROM SMALL DATA: FEATURES, LOCALITY AND DIRECTIONALITY

Magdalena Markowska

DEPARTMENT OF LINGUISTICS STONY BROOK

Stony Brook University

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

NAPhC 2025 @ Concordia

THE SUBREGULAR HYPOTHESIS

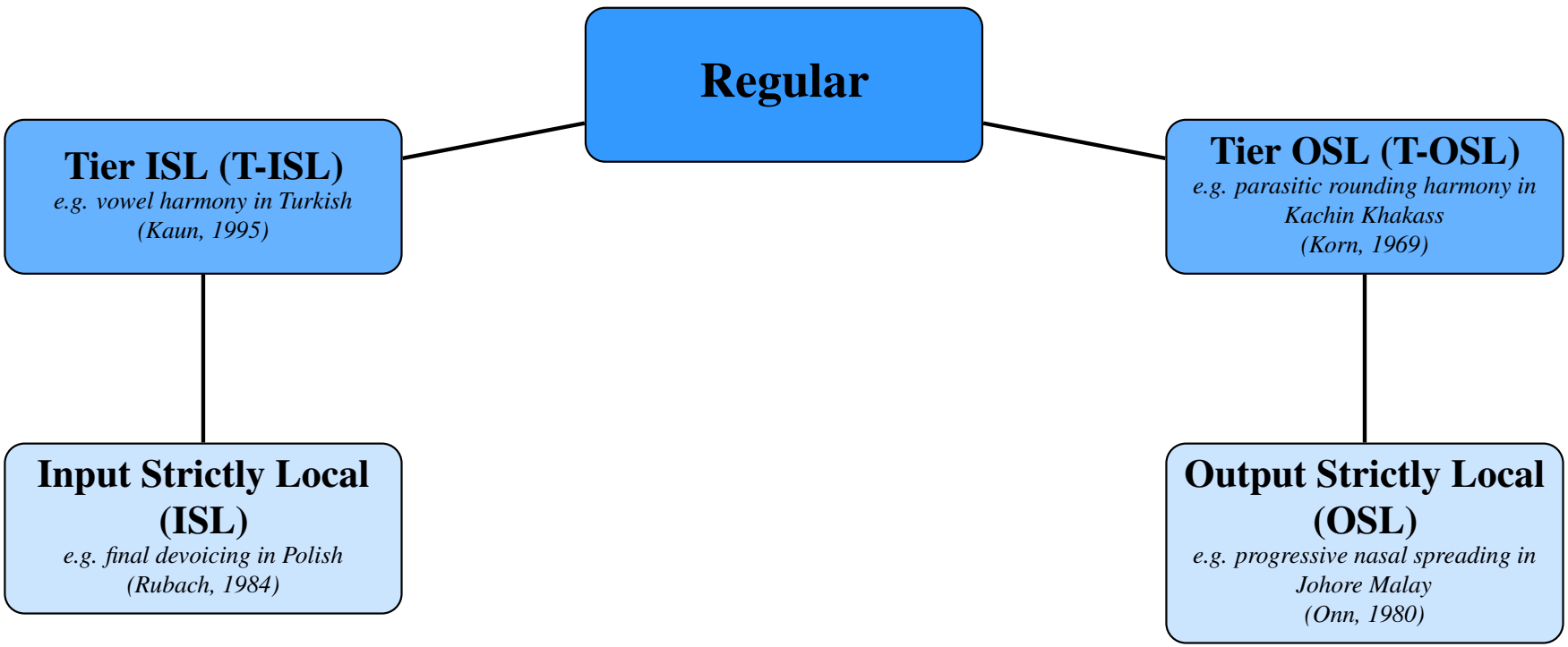
Subregular Phonology seeks to resolve the tension between theories of phonology that are *sufficiently expressive* to account for the cross-linguistic variation in phonological patterning and ones that are *sufficiently constrained* so that grammars can be learned from examples.

For processes these include the input and output strictly local maps and tier-based versions thereof.[1]

[1] Chandlee and Heinz (2018); Burness et al. (2021)

THE SUBREGULAR HYPOTHESIS

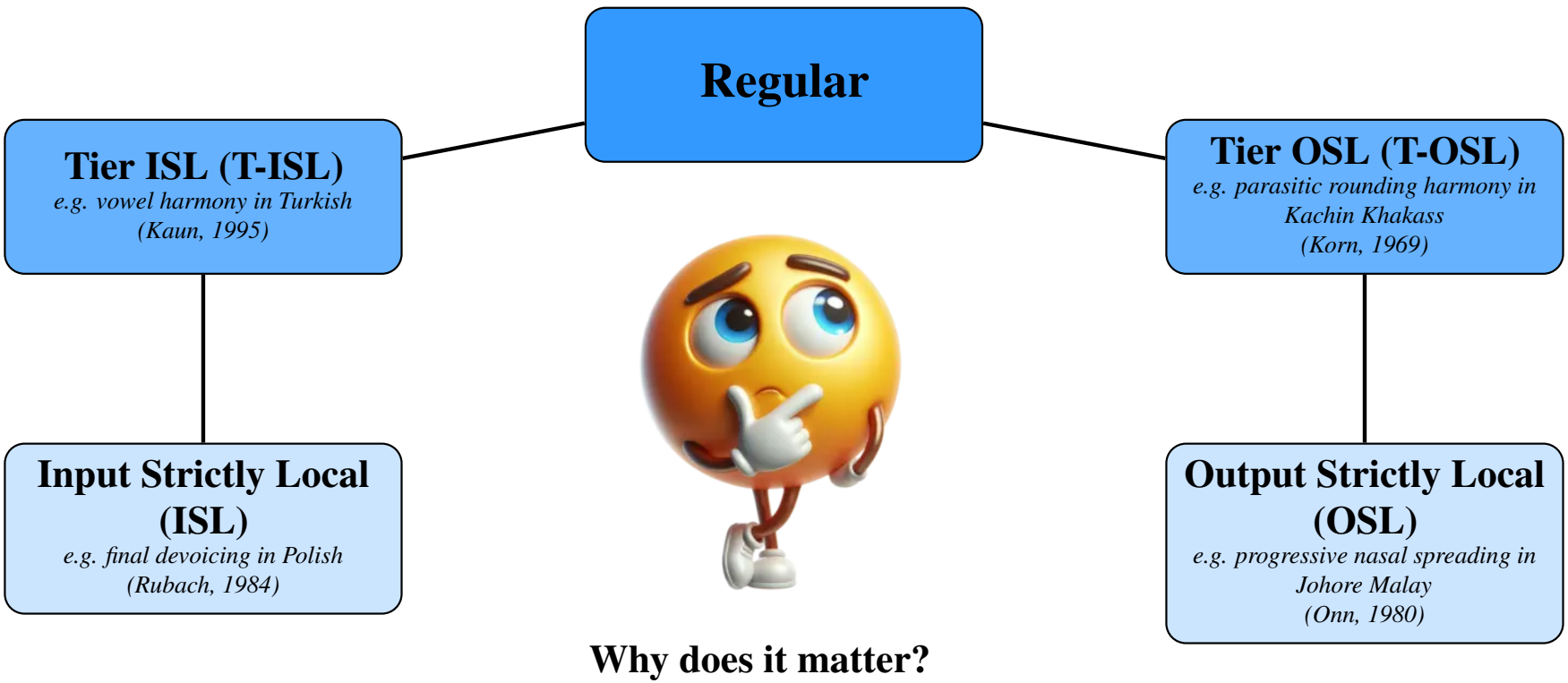
Subregular Phonology argues that most phonological patterns fall within a subregion of regular languages. Multiple classes within this region have been identified[1], for example:



[1] Heinz (2018)

THE SUBREGULAR HYPOTHESIS

Subregular Phonology argues that most phonological patterns fall within a subregion of regular languages. Multiple classes within this region have been identified[1], for example:



Why does it matter?

[1] Heinz (2018)

WHY THE SUBREGULAR HYPOTHESIS MATTERS

1. **Expressivity:**
   Captures both local and non-local phonological patterns[2]
2. **Provable learnability results:**
   Subregular classes are mathematically well-defined, which allows for efficient algorithms that effectively generalize the observed patterns [3]
3. **Computationally tractable:**
   Many patterns can be represented with *subsequential functions*[4] instantiated by *deterministic finite state transducers* (DFTs), making them suitable for various language technology applications

[2] Heinz (2010); Jardine (2016); Heinz (2018); McMullin and Hansson (2019)

[3] Heinz and Rogers (2013); Chandlee (2014); Chandlee et al. (2015a); Jardine and Heinz (2016)

[4] Sakarovitch (2009); Heinz and Lai (2013); Chandlee et al. (2014), Chandlee (2017); Chandlee and Heinz (2018)

THIS WORK

Over the past two decades, theoretical work in subregular phonology has led to the development of various learning **algorithms** that can successfully identify morphophonological patterns in the limit, given a characteristic sample.[5] Here I focus on those that fall within the **ISL** class.

- ISLFLA: $O(n^2)$
- SOSFIA: $O(n)$

[5] Oncina et al. (1993); Oncina and Varó (1996); Chandlee et al. (2015b), Burness and McMullin (2019); Chandlee et al. (2014); Jardine et al. (2014)

THIS WORK

Over the past two decades, theoretical work in subregular phonology has led to the development of various learning **algorithms** that can successfully identify morphophonological patterns in the limit, given a characteristic sample.[5] Here I focus on those that fall within the **ISL** class.

- ISLFLA: $O(n^2)$
- SOSFIA: $O(n)$

Practical Limitation: large amounts of training data.

[5] Oncina et al. (1993); Oncina and Varó (1996); Chandlee et al. (2015b), Burness and McMullin (2019); Chandlee et al. (2014); Jardine et al. (2014)

THIS WORK

Over the past two decades, theoretical work in subregular phonology has led to the development of various learning **algorithms** that can successfully identify morphophonological patterns in the limit, given a characteristic sample.[5] Here I focus on those that fall within the **ISL** class.

- ISLFLA: $O(n^2)$
- SOSFIA: $O(n)$

Practical Limitation: large amounts of training data.

For example:

- a characteristic sample that guarantees success in learning vowel nasalization in English consists of approximately 18,278 UR-SR pairs.

[5] Oncina et al. (1993); Oncina and Varó (1996); Chandlee et al. (2015b), Burness and McMullin (2019); Chandlee et al. (2014); Jardine et al. (2014)

THIS WORK

Over the past two decades, theoretical work in subregular phonology has led to the development of various learning **algorithms** that can successfully identify morphophonological patterns in the limit, given a characteristic sample.[5] Here I focus on those that fall within the **ISL** class.

- ISLFLA: $O(n^2)$
- SOSFIA: $O(n)$

Practical Limitation: large amounts of training data.

For example:

- a characteristic sample that guarantees success in learning vowel nasalization in English consists of approximately 18,278 UR-SR pairs.

  (considering 26 out of 44 phonemes)!

[5] Oncina et al. (1993); Oncina and Varó (1996); Chandlee et al. (2015b), Burness and McMullin (2019); Chandlee et al. (2014); Jardine et al. (2014)

THIS WORK

Over the past two decades, theoretical work in subregular phonology has led to the development of various learning **algorithms** that can successfully identify morphophonological patterns in the limit, given a characteristic sample.[6] Here I focus on those that fall within the **ISL** class.

- ISLFLA: $O(n^2)$
- SOSFIA: $O(n)$

⚠ **Practical Limitation:** large amounts of training data.

For example:

- a characteristic sample that guarantees success in learning vowel nasalization in English consists of approximately 18,278 UR-SR pairs (considering 26 out of 44 phonemes)!

[6] Oncina et al. (1993); Oncina and Varó (1996); Chandlee et al. (2015b), Burness and McMullin Chandlee et al. (2014); Jardine et al. (2014)

A SOLUTION PROPOSAL

In this talk I show:

- how to switch from **segment-based** to **feature-based** representations to significantly reduce the amount of data required for learning.
- why the **directionality** of string processing (left-to-right vs. right-to-left) also plays a crucial role in data efficiency.
- how learning k-values for individual features can help reduce data size.

ROADMAP

1. k-ISL functions (Chandlee, 2014):
   - definition and representation
   - training data requirement that guarantees success –> why so large?
2. The role of features in learning phonological functions
3. Right-to-left vs. left-to-right processing
4. Learning k value for individual features
5. Case studies:
   - vowel nasalization in English (6,157 → **44**)
   - vowel shortening in Yawelmani (16,583 → **164**)
   - ə-epenthesis in Chukchi (17,181 → **2,331**)
   - opaque interaction of final devoicing and *o*-raising in Polish (16,583 → **8,211**)

k-ISL FUNCTIONS AND k-ISL TRANSDUCERS

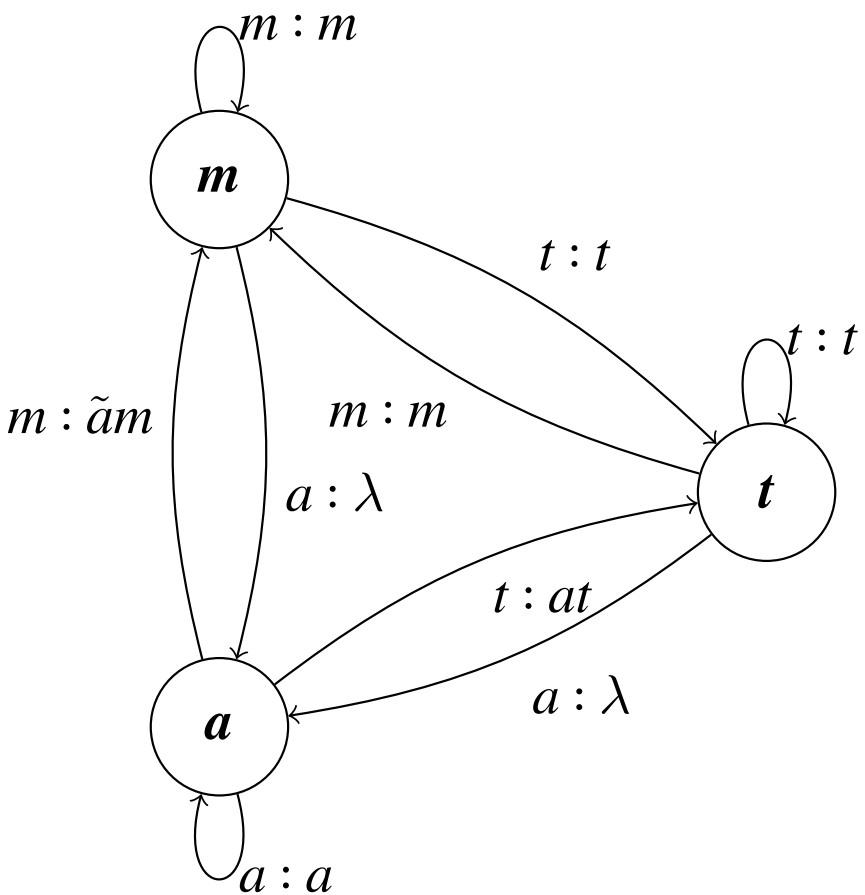
k-ISL functions can be instantiated with a *deterministic finite-state transducer (**DFT**)*, whose states and transitions are organized in such a way that the current state of the machine is always determined by the previous $k-1$ symbols on the input.

Crucially, every k-ISL DFTs has **the same structure**.

- vowel nasalization in English ($VN \rightarrow \tilde{V}N$): $k = 2$
- final devoicing in Polish ($D\ltimes \rightarrow T\ltimes$): $k = 2$

VOWEL NASALIZATION AS A 2-ISL PROCESS

$\Sigma = \{a, m, t\}$
$\Delta = \Sigma \cup \{\tilde{a}\}$

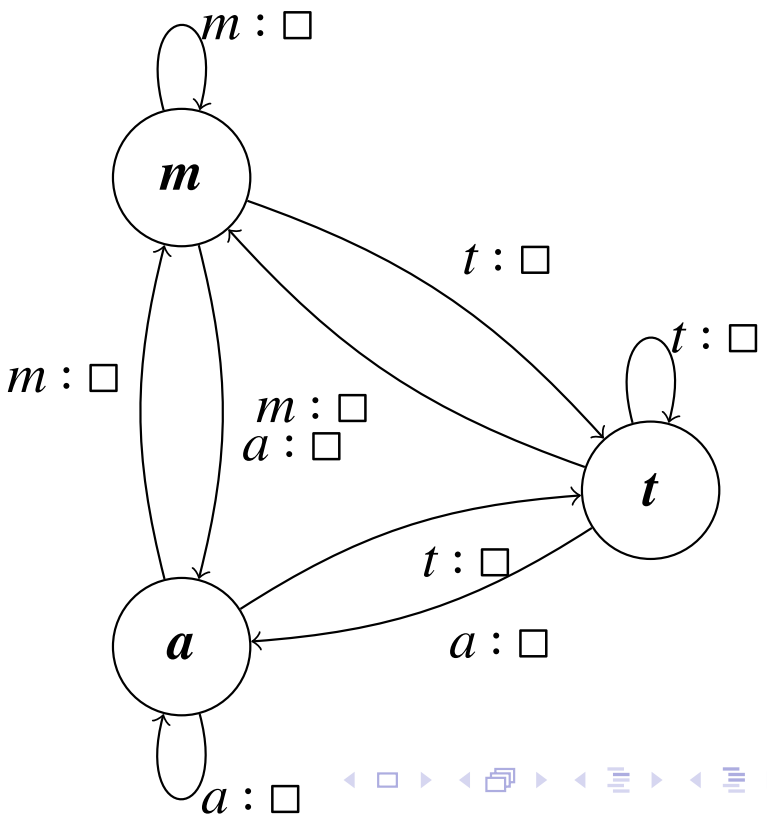


HOW DO WE LEARN SUCH FUNCTIONS?

SOSFIA

Given a characteristic sample of input-output pairs and an output empty DFT for the relevant class, SOSFIA is **guaranteed** to calculate the right outputs, for every transition (i.e. learn the function).

...		...
/kæm/	→	[kæ̃m]
/kkm/	→	[kkm]
/mæm/	→	[mæ̃m]
/mæt/	→	[mæt]
...		...



DATA REQUIREMENTS FOR SOSFIA

The size of k-ISL DFT depends directly on the window size k and the size of the input alphabet Σ.

The state space for such machines can be expresses as $|Q| = |\Sigma|^{k-1}$.

Because SOSFIA needs to see strings that correspond to paths for every transition, the **characteristic sample** S that guarantees success will be, roughly speaking, proportional to $|Q|$. Let's consider some numbers:

	$\|\Sigma\|$	$\|Q\|$	$\|S\|$
	4	4	84
$k = 2$	10	10	1,110
	50	50	127,550
	4	20	1,365
$k = 3$	10	110	111,100
	50	2550	318,877,550

EXAMPLE

Transition: $q = (current\_state, input, output, next\_state)$
$\mathbf{f}_1$: k=2, $V \rightarrow \tilde{V}$ ___ M
$\mathbf{f}_2$: k=3, $V \rightarrow \tilde{V}$ ___ MM

	Transition:	**Required data**:
$\mathbf{f}_1$	(⋉, a, λ, a)	{aa–aa, am–ãm, at–at}
	(a, a, a, a)	{aaa–aaa, aat–aat, aam–aãm}
	(a, m, ãm, m)	{ama–ãma, amt–ãmt, amm–ãmm}
	(a, ⋊, a, ⋊)	{a–a}
	...	...
$\mathbf{f}_2$	(a, a, a, aa)	{aaa–aaa, aam–aãm, aaaa-aaaa, aaam–aaãm, ... }
	(aa, m, ãm, am)	{aama–aãma, aammm–aãmmm, aammt–aãmmt, ...}
	(am, t, t, mt)	{amta–ãmta, amtaa–ãmtaa, amtat–ãmtat ...}
	(mt, ⋊, λ, ⋊)	{mt–mt}
	...	...

SOME ASSUMPTIONS AND OBSERVATIONS

- If no data available for a transition, the output is assumed to be *faithful* to the input.
- For *non-identical* transitions, the learner has to be exposed to very particular strings.
  - **The more non-identical transitions, the more data required.**
- The structure of a k-ISL DFT also directly affects the data size:
  - The bigger the k the bigger the k-ISL DFT.
  - The bigger the alphabet the bigger the k-ISL DFT.
  - **The bigger the k-ISL DFT, the more data the learner needs.**

SOME ASSUMPTIONS AND OBSERVATIONS

- If no data available for a transition, the output is assumed to be *faithful* to the input.
- For *non-identical* transitions, the learner has to be exposed to very particular strings.
  - **The more non-identical transitions, the more data required.**
- The structure of a k-ISL DFT also directly affects the data size:
  - The bigger the k the bigger the k-ISL DFT.
  - The bigger the alphabet the bigger the k-ISL DFT.
  - **The bigger the k-ISL DFT, the more data the learner needs.**

The goal is then to:

I. Reduce the machine size

II. Reduce the amount of non-identical transitions

SOME ASSUMPTIONS AND OBSERVATIONS

- If no data available for a transition, the output is assumed to be *faithful* to the input.
- For *non-identical* transitions, the learner has to be exposed to very particular strings.
  - **The more non-identical transitions, the more data required.**
- The structure of a k-ISL DFT also directly affects the data size:
  - The bigger the k the bigger the k-ISL DFT.
  - The bigger the alphabet the bigger the k-ISL DFT.
  - **The bigger the k-ISL DFT, the more data the learner needs.**

The goal is then to:

I. Reduce the machine size → FEATURES + k-value

II. Reduce the non-identical transitions → DIRECTIONALITY

THE MAIN IDEA

Decompose the learning problem by generalizing over **features**.

- A learner predicts next feature values, as opposed to next symbol.

Let F be a finite set of features with binary or ternary values. If $f \in F$, ϕ_f is a homomorphism from Σ to the set $\{+, -, 0\}$.

- $\phi_{[\text{cor}]}(mat) = [\,-\ 0\ +\,]$
- $\phi_{[\text{cons}]}(mat) = [\,+\ -\ +\,]$

If Φ is an ordered set of features, then Φ is a pointwise product of its homomorphisms.

- $\Phi_{\left[\begin{smallmatrix}\text{cor}\\ \text{cons}\end{smallmatrix}\right]}(mat) = \left[\begin{smallmatrix}-&0&+\\ +&-&+\end{smallmatrix}\right]$

Finally, if S is a sample of input/output pairs composed of segments, $S_{(F,f)}$ is sample for feature f.

- $S_{(F,f)} = \{(\Phi_F(x),\ \Phi_f(y))\,|\,(x,y) \in S\}$

FACTORING BY FEATURE (FXF)

Given a characteristic sample S, for each $f \in F$:

STEP I. Project $S_{([f],f)}$.

STEP II. Check if $S_{([f],f)}$ is *functional*.

A. If yes, given a $\tau_{\square}$, extract a k-ISL DFT_f and calculate the outputs with a learner (e.g. SOSFIA)

B. If not, pick another $f' \in F$ and project $S_{([f,f'],f)}$.

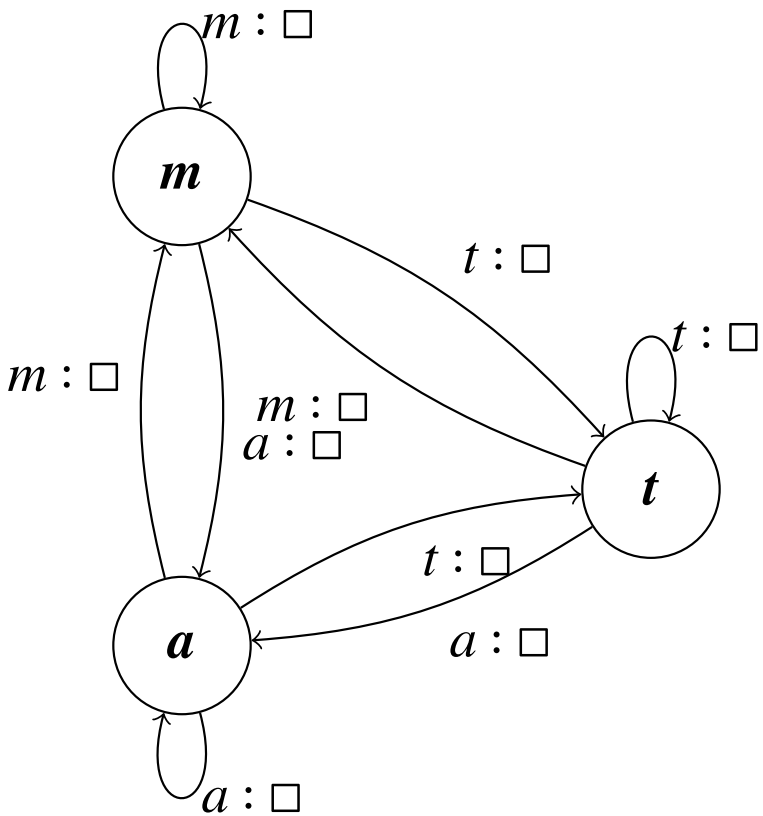
Repeat Step II until a functional sample is obtained.

AN EXAMPLE FROM ENGLISH

$k = 2$

$F = \{\text{cons, nasal, cor, lat,...}\}$

...		...
/kæm/	→	[kæ̃m]
/kkm/	→	[kkm]
/mæn/	→	[mæ̃m]
/mæt/	→	[mæt]
...		...



FEATURE CONSONANTAL

$$k = 2$$

$$f = [\textbf{consonantal}]$$

$$
\begin{array}{ccc}
\ldots & & \ldots \\
/+-+/ & \rightarrow & [+-+] \\
/+++/ & \rightarrow & [+++] \\
/+--/ & \rightarrow & [+--] \\
/---/ & \rightarrow & [---] \\
\ldots & & \ldots
\end{array}
$$

FEATURE CONSONANTAL

$$k = 2$$
$$f = [\textbf{consonantal}]$$

$$\begin{array}{ccc} \cdots & & \cdots \\ /+-+/ & \rightarrow & [+-+] \\ /+++/ & \rightarrow & [+++] \\ /+--/ & \rightarrow & [+--] \\ /---/ & \rightarrow & [---] \\ \cdots & & \cdots \end{array}$$

check_if_functional

FEATURE CONSONANTAL

$$k = 2$$

$$f = [\textbf{consonantal}]$$

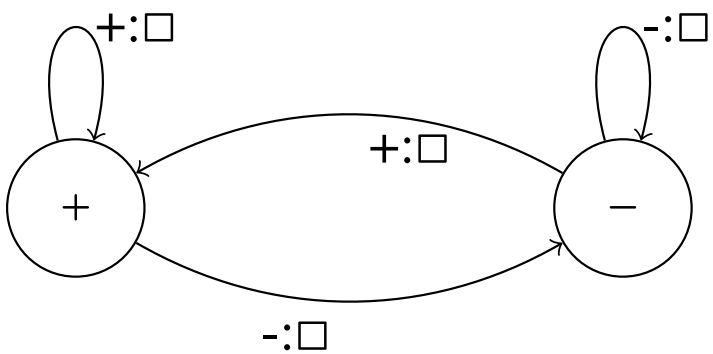
$$\begin{array}{ccc} \ldots & & \ldots \\ /+-+/ & \rightarrow & [+-+] \\ /+++/ & \rightarrow & [+++] \\ /+--/ & \rightarrow & [+--] \\ /---/ & \rightarrow & [---] \\ \ldots & & \ldots \end{array}$$

check_if_functional True

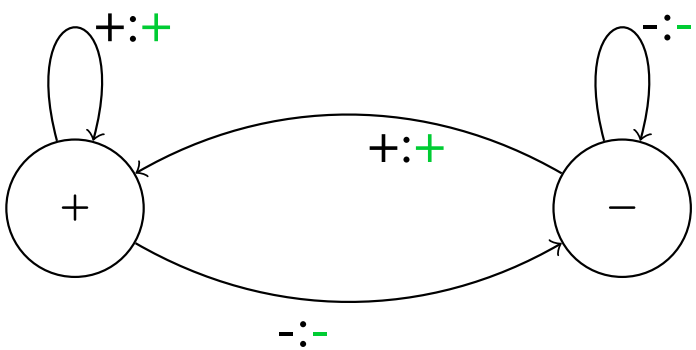
FEATURE CONSONANTAL

$$k = 2$$
$$f = [\textbf{consonantal}]$$

…		…
/+−+/	→	[+−+]
/+++/	→	[+++]
/+−−/	→	[+−−]
/−−−/	→	[−−−]
…		…



SOSFIA FXF'S OUTPUT FOR [CONSONANTAL]



FEATURE NASAL

$$k = 2$$
$$f = [\textbf{nasal}]$$

$$\begin{array}{ccc} \ldots & & \ldots \\ /{-}{-}{+}/ & \rightarrow & [\,{-}{+}{+}\,] \\ /{-}{-}{+}/ & \rightarrow & [\,{-}{-}{+}\,] \\ /{+}{-}{+}/ & \rightarrow & [\,{+}{+}{+}\,] \\ /{+}{-}{+}/ & \rightarrow & [\,{+}{-}{+}\,] \\ \ldots & & \ldots \end{array}$$

FEATURE NASAL

$$k = 2$$
$$f = [\textbf{nasal}]$$

$$
\begin{array}{ccc}
\cdots & & \cdots \\
/--+/ & \rightarrow & [-++] \\
/--+/ & \rightarrow & [--+] \\
/+-+/ & \rightarrow & [+++] \\
/+-+/ & \rightarrow & [+-+] \\
\cdots & & \cdots
\end{array}
$$

check_if_functional

FEATURE NASAL

$$k = 2$$
$$f = [\textbf{nasal}]$$

...		...
/−−+/	→	[−++]
/−−+/	→	[−−+]
/+−+/	→	[+++]
/+−+/	→	[+−+]
...		...

check_if_functional False

FEATURES NASAL AND CONSONANTAL

$k = 2$

f, f' = [**nasal**, consonantal]

...		...
$/\begin{bmatrix}-\\+\end{bmatrix}\begin{bmatrix}-\\-\end{bmatrix}\begin{bmatrix}+\\+\end{bmatrix}/$	$\rightarrow$	$[-++]$
$/\begin{bmatrix}-\\+\end{bmatrix}\begin{bmatrix}-\\+\end{bmatrix}\begin{bmatrix}+\\+\end{bmatrix}/$	$\rightarrow$	$[--+]$
$/\begin{bmatrix}+\\+\end{bmatrix}\begin{bmatrix}-\\-\end{bmatrix}\begin{bmatrix}+\\+\end{bmatrix}/$	$\rightarrow$	$[+++]$
$/\begin{bmatrix}+\\+\end{bmatrix}\begin{bmatrix}-\\-\end{bmatrix}\begin{bmatrix}-\\+\end{bmatrix}/$	$\rightarrow$	$[--+]$
...		...

FEATURES NASAL AND CONSONANTAL

$$k = 2$$

$$f, f' = [\textbf{nasal}, \text{consonantal}]$$

$$\begin{array}{lcl} \cdots & & \cdots \\ / \begin{bmatrix} - \\ + \end{bmatrix} \begin{bmatrix} - \\ - \end{bmatrix} \begin{bmatrix} + \\ + \end{bmatrix} / & \rightarrow & [-++] \\ / \begin{bmatrix} - \\ + \end{bmatrix} \begin{bmatrix} - \\ + \end{bmatrix} \begin{bmatrix} + \\ + \end{bmatrix} / & \rightarrow & [--+] \\ / \begin{bmatrix} + \\ + \end{bmatrix} \begin{bmatrix} - \\ - \end{bmatrix} \begin{bmatrix} + \\ + \end{bmatrix} / & \rightarrow & [+++] \\ / \begin{bmatrix} + \\ + \end{bmatrix} \begin{bmatrix} - \\ - \end{bmatrix} \begin{bmatrix} - \\ + \end{bmatrix} / & \rightarrow & [--+] \\ \cdots & & \cdots \end{array}$$

check_if_functional

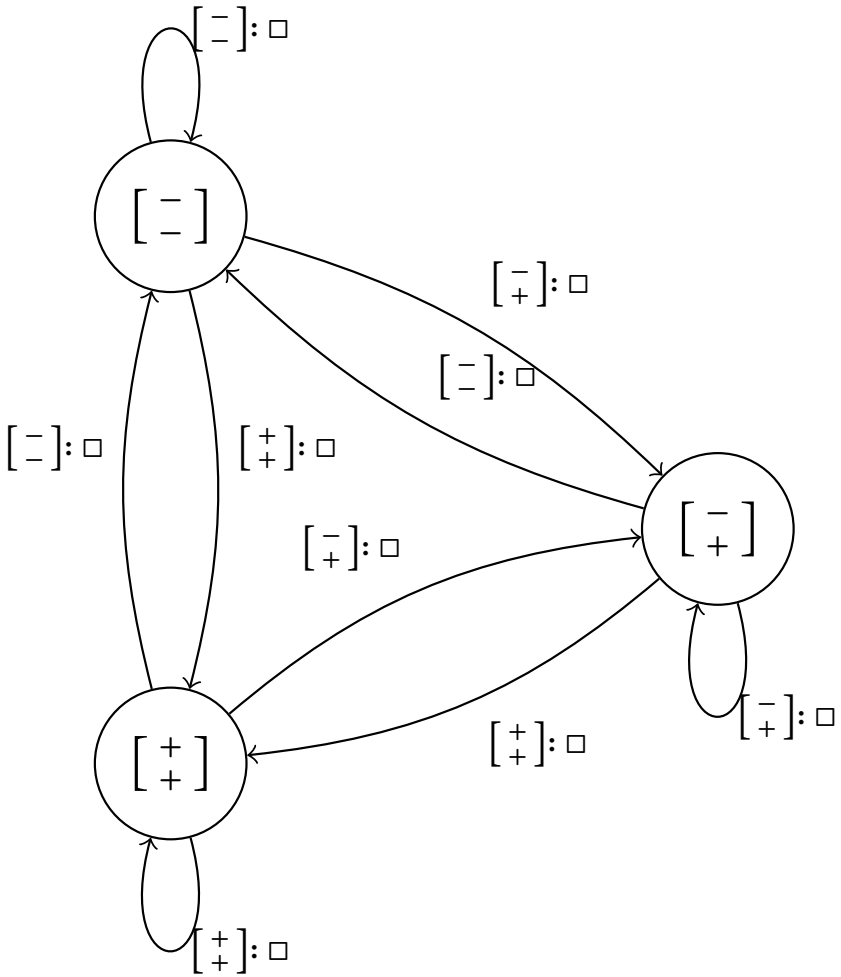
FEATURES NASAL AND CONSONANTAL

$$k = 2$$

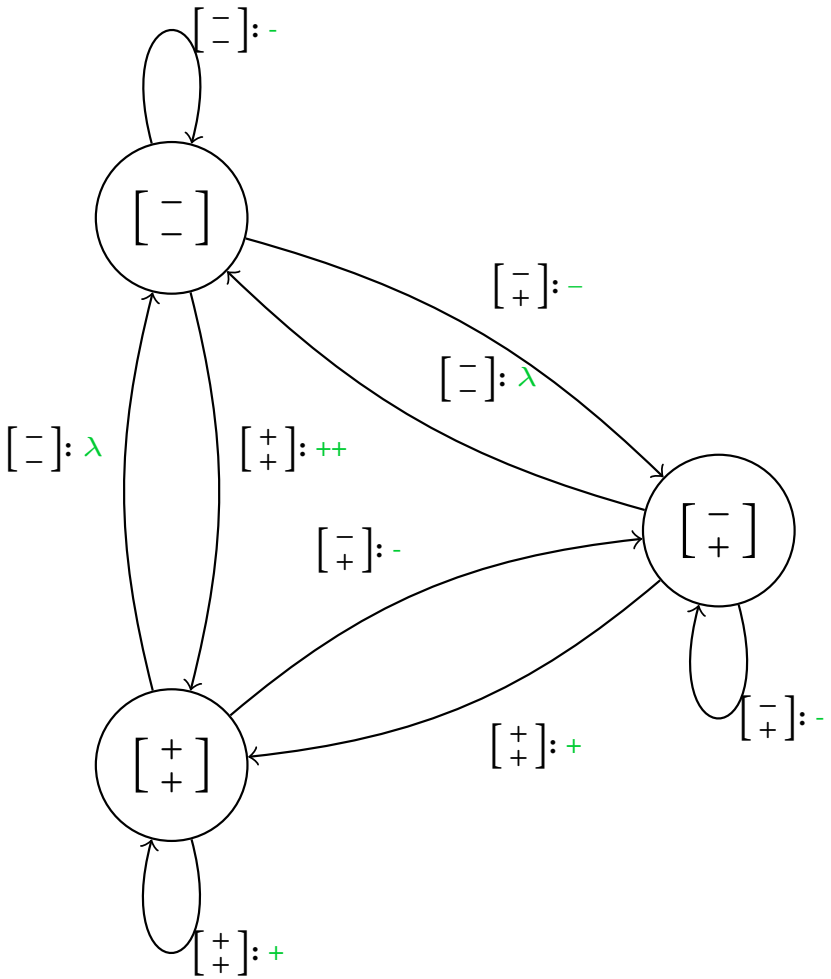
$$f, f' = [\textbf{nasal}, \text{consonantal}]$$

...		...
$/\begin{bmatrix}-\\+\end{bmatrix}\begin{bmatrix}-\\-\end{bmatrix}\begin{bmatrix}+\\+\end{bmatrix}/$	$\rightarrow$	$[-++]$
$/\begin{bmatrix}-\\+\end{bmatrix}\begin{bmatrix}-\\+\end{bmatrix}\begin{bmatrix}+\\+\end{bmatrix}/$	$\rightarrow$	$[--+]$
$/\begin{bmatrix}+\\+\end{bmatrix}\begin{bmatrix}-\\-\end{bmatrix}\begin{bmatrix}+\\+\end{bmatrix}/$	$\rightarrow$	$[+++]$
$/\begin{bmatrix}+\\+\end{bmatrix}\begin{bmatrix}-\\-\end{bmatrix}\begin{bmatrix}-\\+\end{bmatrix}/$	$\rightarrow$	$[+--]$
...		...
check_if_functional		True

OUTPUT EMPTY DFT$_{\left[\begin{smallmatrix} nasal \\ cons \end{smallmatrix}\right]}$



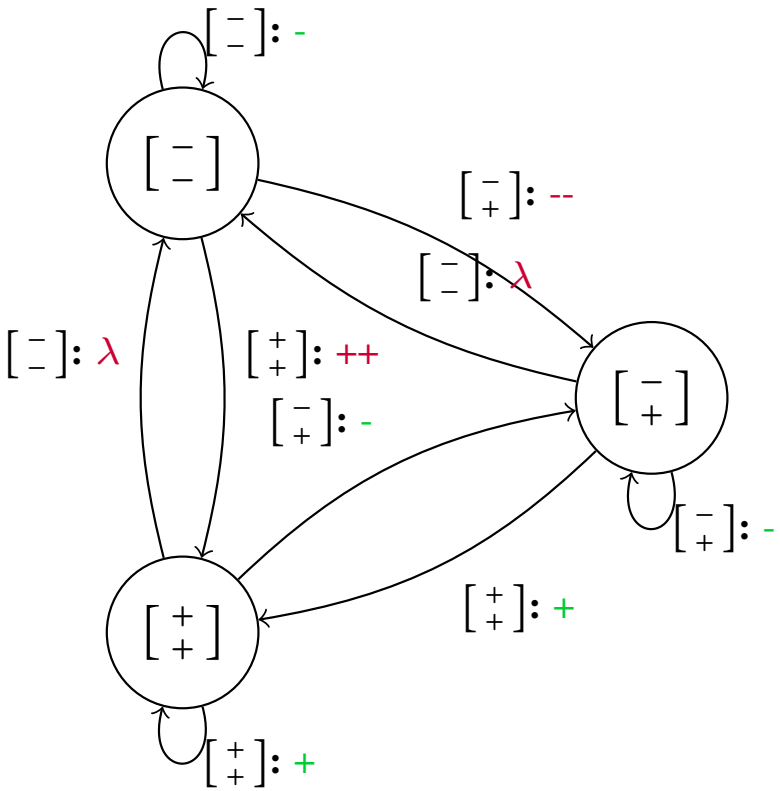
SOSFIA'S OUTPUT FOR $\begin{bmatrix} \textit{nasal} \\ \textit{cons} \end{bmatrix}$



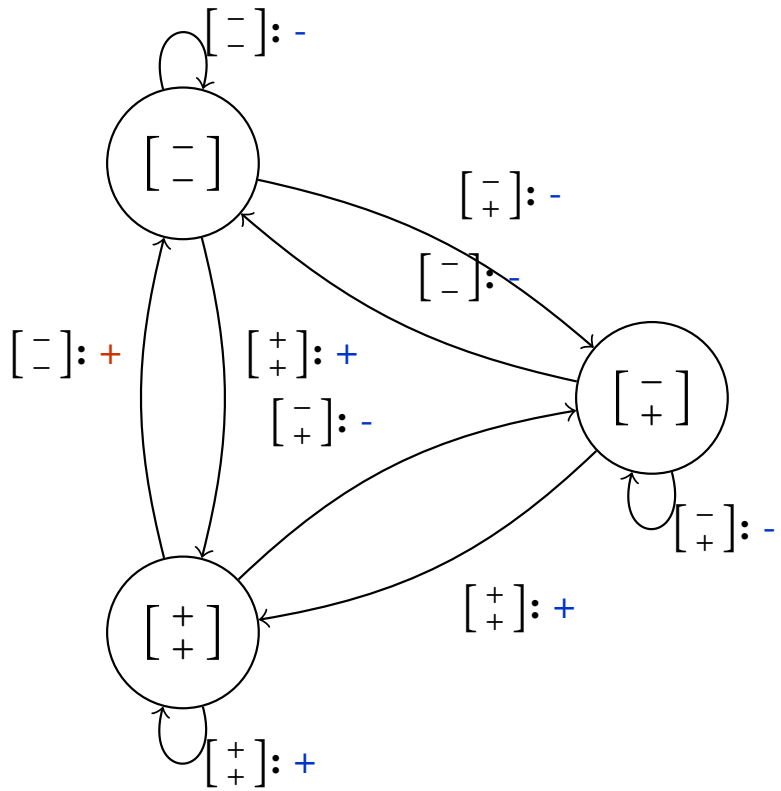
DIRECTIONALITY AND k-ISL FUNCTIONS

Whether a string is processed from left-to-right or right-to-left does not affect the **structure** of the machine (Chandlee, 2014).

left-to-right



right-to-left



DIRECTIONALITY AND k-ISL FUNCTIONS

The comparison from before showed us that changing the direction of the string affects only the outputs: only the transitions leaving nasal consonant state ($[\begin{smallmatrix}+\\+\end{smallmatrix}]$) will be different from the inputs.

Essentially, **changing the direction can reduce** the number of non-identical transitions and thus reduce the **data** needed.

WHICH DIRECTION TO GO?

Changing direction can be achieved by simply reversing the data; the structure of the machine remains unchanged.

But how to know which direction to take?

WHICH DIRECTION TO GO?

Changing direction can be achieved by simply reversing the data; the structure of the machine remains unchanged.

But how to know which direction to take?

- flip a coin

WHICH DIRECTION TO GO?

Changing direction can be achieved by simply reversing the data; the structure of the machine remains unchanged.

But how to know which direction to take?

- flip a coin
- study the data

WHICH DIRECTION TO GO?

Changing direction can be achieved by simply reversing the data; the structure of the machine remains unchanged.

But how to know which direction to take?

- flip a coin
- study the data

Given the data, consider only the unfaithful input-output pairs and calculate the longest common prefix (`lcp`) for all substrings starting from the edge $\rtimes$.

- Change directionality if for any substring $\rtimes\sigma^{+}, \sigma \in \Sigma$ `lcp` $= \lambda$

HOW CAN k-VALUE BE LEARNED?

As many phonological processes are *feature-filling* or *feature-changing*, there often exists a subset of features that do not undergo any changes.

HOW CAN k-VALUE BE LEARNED?

As many phonological processes are *feature-filling* or *feature-changing*, there often exists a subset of features that do not undergo any changes.

This observation allows us for a possibility to adjust the memory window, k, for each feature individually.

HOW CAN k-VALUE BE LEARNED?

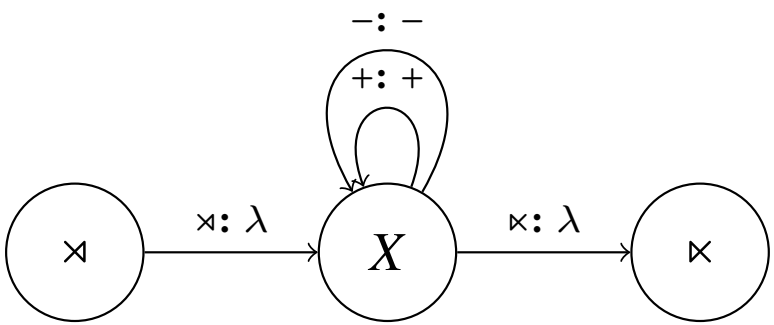
As many phonological processes are *feature-filling* or *feature-changing*, there often exists a subset of features that do not undergo any changes.

This observation allows us for a possibility to adjust the memory window, k, for each feature individually.

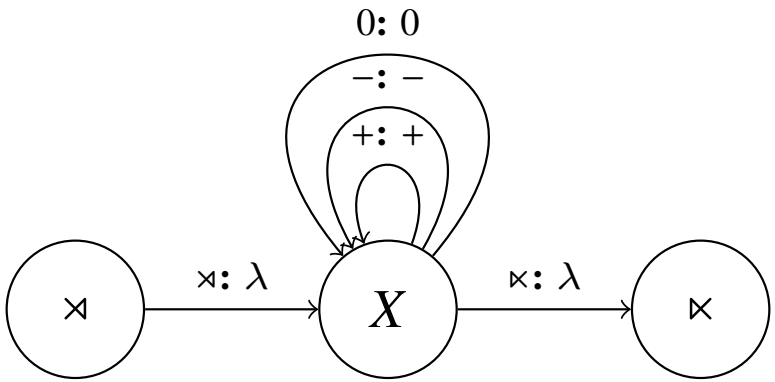
As a result, functions for the faithful features can be represented and learned from **one-state** machines.

AN EXAMPLE FROM ENGLISH

$f = [\textbf{voice}]$

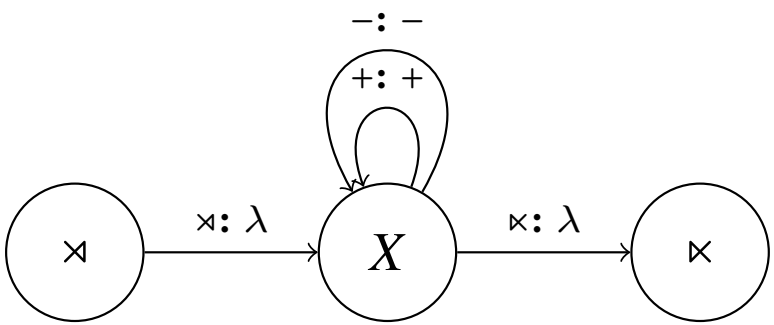


$f = [\textbf{coronal}]$

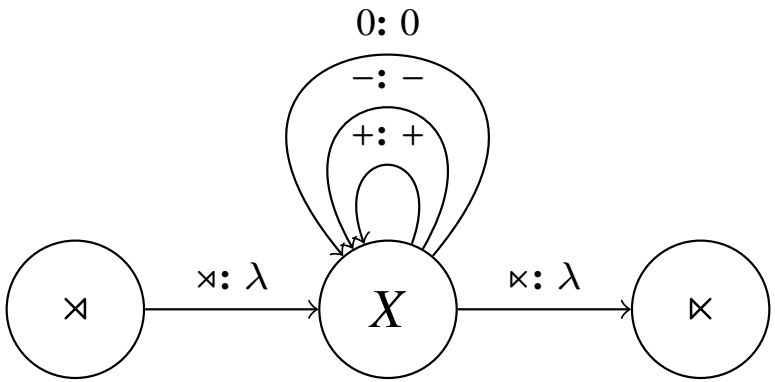


AN EXAMPLE FROM ENGLISH

$f = [\textbf{voice}]$



$f = [\textbf{coronal}]$



Regardless of the number of feature values, the size of 1-ISL machine will remain the same.

PUTTING EVERYTHING TOGETHER

The notions of features, locality, and directionality create multiple possibilities.

The goal of a learner then is to start with the simplest assumption, i.e. a feature can be learned from itself with no memory stored ($k = 1$). If unsuccessful, the learner has three options:

1. add another feature to the input
2. increase k
3. reverse the direction

PUTTING EVERYTHING TOGETHER

The notions of features, locality, and directionality create multiple possibilities.

The goal of a learner then is to start with the simplest assumption, i.e. a feature can be learned from itself with no memory stored ($k = 1$). If unsuccessful, the learner has three options:

1. add another feature to the input
2. increase k
3. reverse the direction

Soon coming 4th dimension: **tiers**

TARGET k-ISL FUNCTIONS

- **Vowel Nasalization in English** (2-ISL; 22 features)
  $\left[\text{- cons} \right] \longrightarrow \left[\text{+ nasal} \right] / \_\_\_\_ \left[\begin{matrix} \text{+ cons} \\ \text{- nasal} \end{matrix} \right]$

- **Vowel Shortening in Yawelmani** (3-ISL; 22 features)
  $\left[\text{- cons} \right] \longrightarrow \left[\text{- long} \right] / \_\_\_\_ \left[\text{+ cons} \right] \left[\text{+ cons} \right]$

- **ə-epenthesis in Chukchi** (3-ISL; 22 features)
  $\emptyset \longrightarrow$ ə $/ \left[\text{+ cons} \right] \_\_\_\_ \left[\text{+ cons} \right] \ltimes$

- **Final Devoicing and o-raising in Polish** (3-ISL; 17 features)
  $\left[\text{- son} \right] \longrightarrow \left[\text{- voice} \right] / \_\_\_\_ \ltimes$
  $\left[\begin{matrix} \text{- cons} \\ \text{+ back} \\ \text{- low} \end{matrix} \right] \longrightarrow \left[\text{+ high} \right] / \_\_\_\_ \left[\begin{matrix} \text{+ cons} \\ \text{+ voice} \\ \text{- nasal} \end{matrix} \right] \ltimes$

ALPHABET AND CHARACTERISTIC SAMPLE

- 2-ISL function:
  - $|\Sigma| = 26$
  - $|S| = 18,278$
- 3-ISL function:
  - $|\Sigma| = 7$
  - $|S| = 19,607$

EXPERIMENTS

Each function is learned:

- over **segments**, tested for learnability
- separately for each **feature** $f \in F$, also tested for success
- in both **left-to-right** and **right-to-left** processing directions
- $k = 1$ by default and increased when necessary

We also introduce a minimal sample search procedure: **AMβA**, which identifies a minimal subset of S sufficient for learning. We draw batches without replacement and test for success.

This yields two sample types:

- M: **segmental sample**, batch size = 50
- $M_{(F,f)}$: **featural sample** for feature f, batch size = 1
- We run **10 iterations** and report the average and standard deviation (SD)

Finally, we compute a **synthesis** of all individual featural samples to allow direct comparison with segmental performance.

ENGLISH: RESULTS

Feature f	Left-to-Right				Right-to-Left			
	Set $k = 2$		Learned k		Set $k = 2$		Learned k	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
$[\,^{\textbf{nas}}_{cons}\,]$	31	3.88	29	10.10	32	4.33	27	3.90
[round]	8	2.27	3	0	6	3.57	3	0
[cor]	23	4.76	4	0	16	11.73	4	0
...	...	...	...	...	...	...	...	...
Synthesis (Avg.)	**189**	10.06	**46**	3.71	**137**	10.50	**44**	1.89
Segments (Avg. $\lvert M \rvert$)	**6,157**	650.21	–	–	–	–	–	–
$\lvert S \rvert$	18,278							

YAWELMANI: RESULTS

Feature f	Left-to-Right				Right-to-Left			
	Set $k = 3$		Learned k		Set $k = 3$		Learned k	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
[high]	148	27.12	4	0	147	23.42	4	0
[long]	317	49.02	251	63.26	169	17.58	155	21.02
[cons]	36	5.47	3	0	30	11.52	3	0
...	...	...	...	...	...	...	...	...
Synthesis (Avg.)	**952**	49.90	**258**	62.95	**851**	56.51	**164**	21.24
Segments (Avg. $\lvert M \rvert$)	**16,583**	702.06	–	–	–	–	–	–
$\lvert S \rvert$	18,278							

Chukchi: Results

Feature f	$\lvert M_{(F,f)} \rvert$ Left-to-Right Avg.	Left-to-Right SD	Right-to-Left Avg.	Right-to-Left SD
[**round**]	165	30.34	192	51.10
[**son** *low*]	225	30.28	227	35.57
[**cont** *tense*]	260	12.77	244	19.54
[**delrel** *long*]	924	57.82	892	41.75
[**distr** *cor*]	801	14.98	917	55.02
[**tense**]	32	5.69	36	9.56
...	...	...	...	...
Synthesis (Avg.)	**2,331**	46.27	**2,352**	43.24
Segments (Avg. |M|)	**17,181**	28.87	–	–
|S|		19,607		

POLISH: RESULTS

Feature f	Left-to-Right				Right-to-Left			
	Set $k = 3$		Learned k		Set $k = 3$		Learned k	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
[**voice** son]	214	10.73	193	29	233	85.93	101	38.13
[**high** voice son low]	7,119	907.70	6,984	1,262.72	7,360	630.27	7,156	695.43
[**round**]	137	51.76	4	0	173	45.04	4	0
[**dor**]	32	4.33	3	0	31	11.15	3	0
...	...	...	...	...	...	...	...	...
Synthesis (Avg.)	**8,235**	208.39	**8,211**	139.89	**8,114**	145.28	**8,068**	202.83
Segments (Avg. $\lvert M \rvert$)	**16,583**	702.06	–	–	–	–	–	–
$\lvert S \rvert$	19,607							

POLISH: BIGGER ALPHABET

	$\lvert M_{(F,f)} \rvert$					
	Left-to-Right			**Right-to-Left**		
Feature f	$\Sigma = 8$	$\Sigma = 9$	$\Sigma = 10$	$\Sigma = 8$	$\Sigma = 9$	$\Sigma = 10$
$\left[\begin{smallmatrix}\textbf{voice}\\ son\end{smallmatrix}\right]$	215	239	179	76	126	–
$\left[\begin{smallmatrix}\textbf{high}\\ voice\\ son\\ low\end{smallmatrix}\right]$	7,506	–	–	8,186	–	–
$\left[\begin{smallmatrix}\textbf{high}\\ voice\\ nasal\\ low\end{smallmatrix}\right]$	–	8,827	8,007	–	8,982	–
[**round**]	137	173	134	144	158	–
[**dor**]	35	29	26	35	41	–
...	...	...	...	...	...	...
Synthesis (Avg.)	**8,225**	**8,851**	**8,911**	**8,844**	**8,823**	–
Segments (Avg. $\lvert M \rvert$)	**30,344**	**52,161**	**79,540**	–	–	–
|S|	37,448	66,429	111,110	37,448	66,429	111,110

CONCLUSION

Key findings

- Learning phonological patterns is sensitive to how we represent the data — switching from segments to **features** drastically reduces data requirements.
- Features allow for flexible k values and help to keep track of what is changing exactly and what remains the same.
- **Directionality matters**: in many cases, right-to-left processing leads to smaller and more efficient models.
- A sample search algorithm (**AMβA**,) successfully identifies a smaller sufficient sample. (We can definitely do better!)
- These results offer a roadmap for building interpretable, data-efficient learning models grounded in phonological structure.

FUTURE STEPS

1. Evaluate the approach on more **naturalistic datasets**, including textbook-style data sets
2. Extend the analysis to other classes: **TISL, OSL, TOSL**
3. Design a **searchable model space** with controlled variables — k, features, directionality, tiers — to identify the most data-efficient configurations
4. Incorporate **noisy and incomplete data** to evaluate robustness
5. Integrate **learning lexicon**


THANK
YOU

REFERENCES I

Burness, P. and McMullin, K. (2019). Efficient learning of output tier-based strictly 2-local functions. In de Groote, P., Drewes, F., and Penn, G., editors, Proceedings of the 16th Meeting on the Mathematics of Language, pages 78–90, Toronto, Canada. Association for Computational Linguistics.

Burness, P. A., McMullin, K. J., and Chandlee, J. (2021). Long-distance phonological processes as tier-based strictly local functions. Glossa: a journal of general linguistics, 6(1).

Chandlee, J. (2014). Strictly Local Phonological Processes. PhD thesis, University of Delaware.

Chandlee, J. (2017). Computational locality in morphological maps. Morphology, 27(4):599–641.

Chandlee, J., Eyraud, R., and Heinz, J. (2014). Learning strictly local subsequential functions. Transactions of the Association for Computational Linguistics, 2:491–504.

Chandlee, J., Eyraud, R., and Heinz, J. (2015a). Input strictly local functions. Transactions of the Association for Computational Linguistics, 3:117–128.

Chandlee, J., Eyraud, R., and Heinz, J. (2015b). Output strictly local functions. In Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015), pages 112–125, Chicago, USA. Association for Computational Linguistics.

Chandlee, J. and Heinz, J. (2018). Strict locality and phonological maps. Linguistic Inquiry, 49(1):23–60.

Heinz, J. (2010). Learning long-distance phonotactics. Linguistic Inquiry, 41(4):623–661.

Heinz, J. (2018). The computational nature of phonological generalizations. In Hyman, L. and Plank, F., editors, Phonological Typology, Phonetics and Phonology, chapter 5, pages 126–195. De Gruyter Mouton.

Heinz, J. and Lai, R. (2013). Vowel harmony and subsequentiality. In Kornai, A. and Kuhlmann, M., editors, Proceedings of the 13th Meeting on Mathematics of Language, Sofia, Bulgaria.

REFERENCES II

Heinz, J. and Rogers, J. (2013). Learning subregular classes of languages with grammatical inference. In Proceedings of the 14th Meeting on the Mathematics of Language (MoL), pages 1–11.

Jardine, A. (2016). Locality and non-locality in tonal patterns. PhD thesis, University of Delaware.

Jardine, A., Chandlee, J., Eyraud, R., and Heinz, J. (2014). Very efficient learning of structured classes of subsequential functions from positive data. In Clark, A., Kanazawa, M., and Yoshinaka, R., editors, Proceedings of the Twelfth International Conference on Grammatical Inference (ICGI 2014), volume 34, pages 94–108. JMLR: Workshop and Conference Proceedings.

Jardine, A. and Heinz, J. (2016). Learning tier-based strictly local languages. Transactions of the Association for Computational Linguistics, 4:87–98.

Kaun, A. C. (1995). The typology of rounding harmony: An optimality theoretic approach. Ph.d. dissertation, UCLA.

Korn, E. (1969). Dinka vowel mutation and phonology. Studies in African Linguistics, 1(1):55–72.

McMullin, K. and Hansson, G. (2019). Inductive learning of locality relations in segmental phonology. Phonology, 36(2):345–382.

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

Oncina, J. and Varó, M. A. (1996). Using domain information during the learning of a subsequential transducer. Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence, pages 313–325.

Onn, F. M. (1980). Aspects of Malay phonology and morphology: A generative approach. Ph.d. dissertation, University of Illinois at Urbana-Champaign.

Rubach, J. (1984). Cyclic and lexical phonology. De Gruyter Mouton, Berlin, New York.

Sakarovitch, J. (2009). Elements of Automata Theory. Cambridge University Press. Translated by Reuben Thomas from the 2003 edition published by Vuibert, Paris.