A Non-Parametric Model for the Discovery of Inflectional Paradigms from Plain Text using Graphical Models over Strings

Markus Dreyer
Center for Language and Speech Processing (CLSP)
Human Language Technology Center of Excellence (HLTCOE)
Johns Hopkins University (JHU)

Dissertation Defense, November 5, 2010

Motivation

- Rich morphology

English text

break

break

jump break

jump break

break

break

break

German text

brichst

brecht

springst brechen

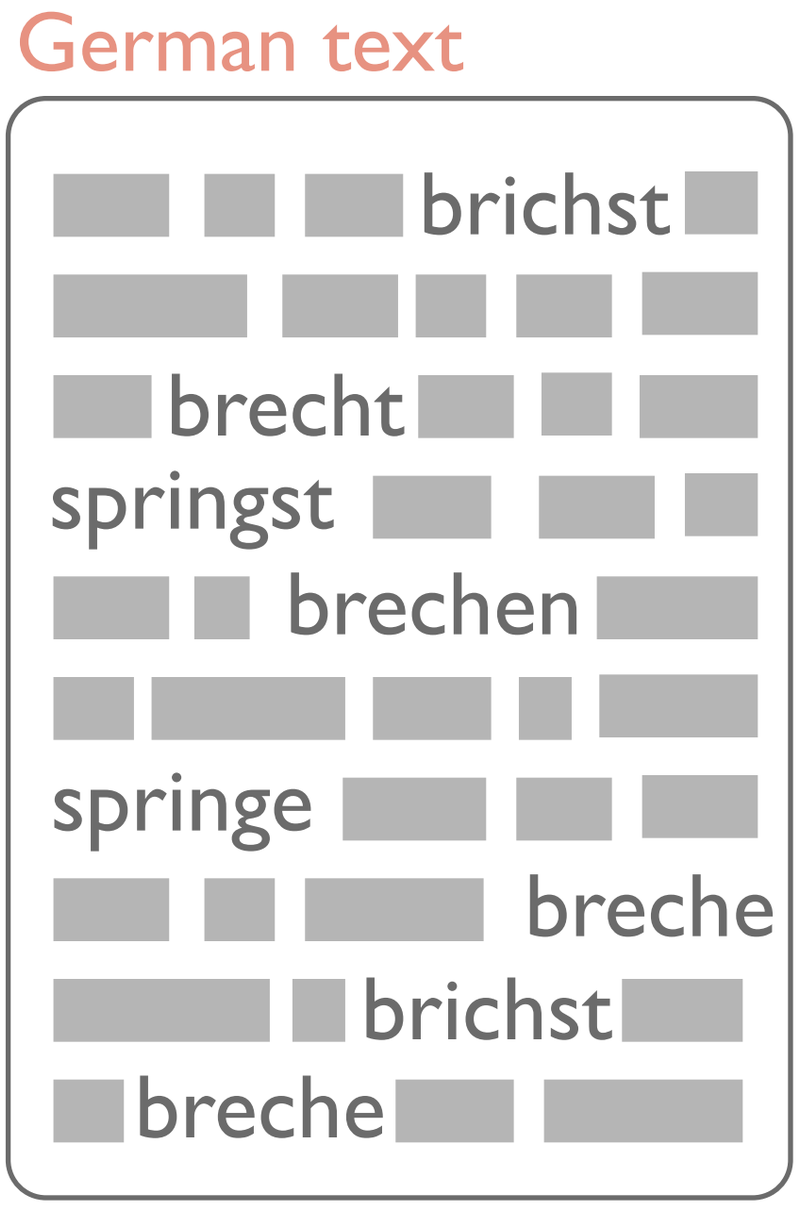
springe breche

brichst

breche

Motivation

- **Analyzing** text:
  - lack of generalization
  - data sparseness
- **Generating** text:
  - need to generate correct forms
  - produce correctly inflected text



Motivation

- In NLP, we often **need** to analyze or generate text in such languages.
- So there is a need for a general **morphology model** that knows **how to inflect words**.
- Since annotations are always expensive, it would be best to **learn from unannotated text**.

Motivation

So how do you inflect a verb?
You look it up in such a table, for example:

Inflectional Paradigm

infinitive	treffen			
1st	treffe	treffen	traf	trafen
2nd	triffst	trefft	trafst	traft
3rd	trifft	treffen	traf	trafen
	singular	plural	singular	plural
	present		past	

But creating such supervised data is **expensive**.

Motivation

- This talk is about a comprehensive **model for inflectional morphology**.
- **Main goal:**
  - Given some **unannotated text**, can we **learn how to inflect** the verbs of a language (incl. irregularities and exceptions)?
  - Discover the **inflectional paradigms** (tables) of a language, using minimal supervision

Motivation

1. Identify the different lexemes in text

German text

brichst

brecht

springst

brechen

springe

breche

brichst

breche

Paradigm

infinitive				
1st				
2nd				
3rd				
	singular	plural	singular	plural
	present		past	

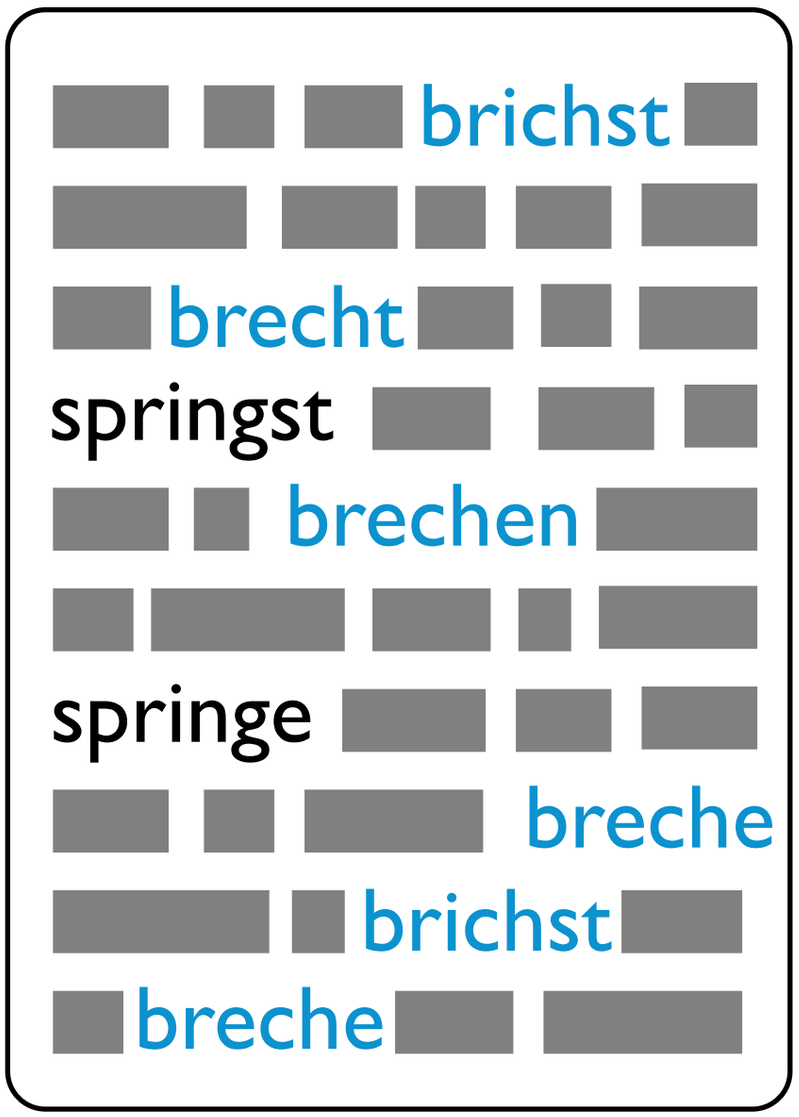
Tokens

Types

Motivation

1. Identify the different lexemes in text

German text



Paradigm

infinitive				
1st				
2nd				
3rd				
	singular	plural	singular	plural
	present		past	

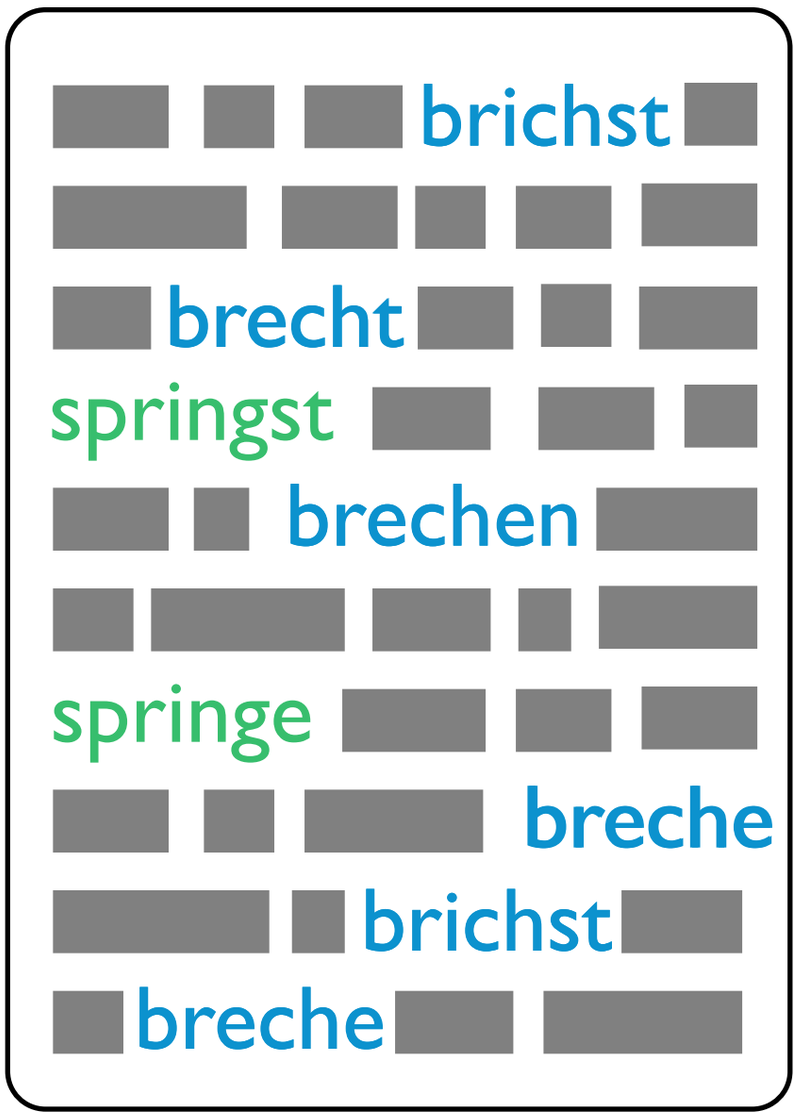
Tokens

Types

Motivation

1. Identify the different lexemes in text

German text



Paradigm

infinitive				
1st				
2nd				
3rd				
	singular	plural	singular	plural
	present		past	

Tokens

Types

Motivation

2. Place each form of a lexeme into its paradigm

German text

brichst

brecht

springst

brechen

springe

breche

brichst

breche

Paradigm

infinitive	brechen			
1st	breche			
2nd	brichst	brecht		
3rd				
	singular	plural	singular	plural
	present		past	

Tokens

Types

Motivation

2. Sort each lexeme into a paradigm

German text

brichst

brecht

springst

brechen

springe

breche

brichst

breche

Paradigm

infinitive	brechen			
1st	breche	brichen? brechen?	brichte? brach?	brichten? brachen?
2nd	brichst	brecht	brichtest? brachst?	brichtet? bracht?
3rd	bricht? brecht?	brichen? brechen?	brichte? brach?	brichten? brachen?
	singular	plural	singular	plural
	present		past	

Tokens

Types

Motivation

2. Sort each lexeme into a paradigm

German text

brichst

brecht

springst

brechen

springe

breche

brichst

breche

Paradigm

infinitive				
1st				
2nd				
3rd				
	singular	plural	singular	plural
	present		past	

Tokens

Types

Motivation

2. Sort each lexeme into a paradigm

German text

brichst

brecht

springst

brechen

springe

breche

brichst

breche

Tokens

Paradigm

infinitive				
1st	springe			
2nd	springst			
3rd				
	singular	plural	singular	plural
	present		past	

Types

Motivation

2. Sort each lexeme into a paradigm

German text

brichst

brecht

springst

brechen

springe

breche

brichst

breche

Paradigm

infinitive	springen? sprengen?			
1st	springe	springen? sprengen?	springte? sprang?	springte? sprang?
2nd	springst	springt? sprengt?	springtest? sprangst?	springtet? sprangt?
3rd	springt? sprengt?	springen? sprengen?	springte? sprang?	springten? sprangen?
	singular	plural	singular	plural
	present		past	

Tokens

Types

Motivation

German text

brichst

brecht

springst

brechen

springe

breche

brichst

breche

Paradigm

infinitive	springen? sprengen?			
1st	springe	springen? sprengen?	springte? sprang?	springte? sprang?
2nd	springst	springt? sprengt?	springtest? sprangst?	springtet? sprangt?
3rd	springt? sprengt?	springen? sprengen?	springte? sprang?	springten? sprangen?
	singular	plural	singular	plural
	present		past	

Tokens

Types

Motivation

German text

brichst
brecht
springst
brechen
springe
breche
brichst
breche

Tokens

Paradigm

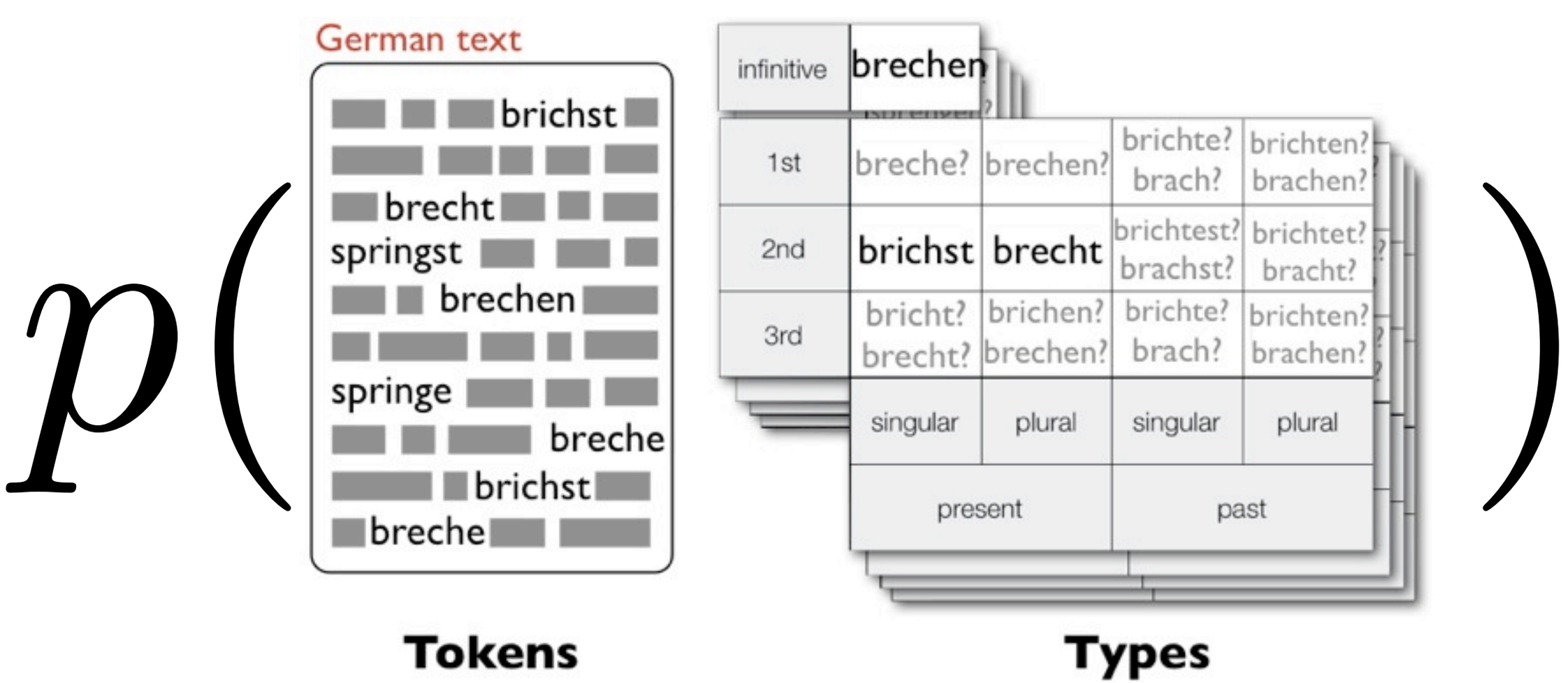
infinitive	saufen	
1st	saufe	säufe saufe
2nd	säufst	sauft
3rd	säuft?	säufe

Motivation

- Similar to **information extraction** tasks:
  - Find information in **text**,
  - Put it in **database**,
  - Make **deductions**,
  - Find **more** information in text,
  - *iterate* ...

Motivation

In order to perform this morphological knowledge discovery, we define a **probability distribution** over a **text corpus** and its (hidden) inflectional **paradigms**:



Tokens **Types**

Overview

1. String pairs

$$p\left(\begin{array}{|c|c|}\hline \text{infinitive} & \textbf{brechen} \\ \hline \text{1st} & \text{breche?} \\ \hline \end{array}\right)$$

2. Multiple strings (paradigms)

$$p\left(\begin{array}{|c|c|c|c|c|}\hline \text{infinitive} & \multicolumn{4}{c|}{\textbf{brechen}} \\ \hline \text{1st} & \text{breche?} & \text{brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \text{2nd} & \textbf{brichst} & \textbf{brecht} & \text{brichtest? brachst?} & \text{brichtet? bracht?} \\ \hline \text{3rd} & \text{bricht? brecht?} & \text{brichen? brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline & \text{singular} & \text{plural} & \text{singular} & \text{plural} \\ \hline & \multicolumn{2}{c|}{\text{present}} & \multicolumn{2}{c|}{\text{past}} \\ \hline \end{array}\right)$$

3. Text and paradigms

$$p\left(\begin{array}{c} \text{German text} \\ \boxed{\begin{array}{l} \blacksquare\blacksquare\blacksquare\textbf{brichst}\blacksquare \\ \blacksquare\blacksquare\blacksquare\blacksquare\blacksquare \\ \blacksquare\textbf{brecht}\blacksquare\blacksquare\blacksquare \\ \textbf{springst}\blacksquare\blacksquare\blacksquare \\ \blacksquare\blacksquare\textbf{brechen}\blacksquare \\ \blacksquare\blacksquare\blacksquare\blacksquare\blacksquare \\ \textbf{springe}\blacksquare\blacksquare\blacksquare \\ \blacksquare\blacksquare\blacksquare\textbf{breche} \\ \blacksquare\blacksquare\textbf{brichst}\blacksquare \\ \blacksquare\textbf{breche}\blacksquare\blacksquare \end{array}} \\ \textbf{Tokens} \end{array}\right)$$

$$\left(\begin{array}{|c|c|c|c|c|}\hline \text{infinitive} & \multicolumn{4}{c|}{\textbf{brechen}} \\ \hline \text{1st} & \text{breche?} & \text{brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \text{2nd} & \textbf{brichst} & \textbf{brecht} & \text{brichtest? brachst?} & \text{brichtet? bracht?} \\ \hline \text{3rd} & \text{bricht? brecht?} & \text{brichen? brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline & \text{singular} & \text{plural} & \text{singular} & \text{plural} \\ \hline & \multicolumn{2}{c|}{\text{present}} & \multicolumn{2}{c|}{\text{past}} \\ \hline \end{array}\right)$$

Types

Overview

1. String pairs

Dreyer, Smith & Eisner, 2008

$$p\left(\begin{array}{|l|l|}\hline \text{infinitive} & \textbf{brechen} \\ \hline \text{1st} & \text{breche?} \\ \hline \end{array}\right)$$

2. Multiple strings (paradigms)

Dreyer & Eisner, 2009

$$p\left(\begin{array}{|l|l|l|l|l|}\hline \text{infinitive} & \textbf{brechen} & & & \\ \hline \text{1st} & \text{breche?} & \text{brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \text{2nd} & \textbf{brichst} & \textbf{brecht} & \text{brichtest? brachst?} & \text{brichtet? bracht?} \\ \hline \text{3rd} & \text{bricht? brecht?} & \text{brichen? brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline & \text{singular} & \text{plural} & \text{singular} & \text{plural} \\ \hline & \multicolumn{2}{|c|}{\text{present}} & \multicolumn{2}{c|}{\text{past}} \\ \hline \end{array}\right)$$

3. Text and paradigms

Dreyer & Eisner, *in prep*

$$p\left(\begin{array}{|l|}\hline \text{German text} \\ \hline \blacksquare\blacksquare\blacksquare\textbf{brichst}\blacksquare \\ \blacksquare\blacksquare\blacksquare\blacksquare\blacksquare \\ \blacksquare\textbf{brecht}\blacksquare\blacksquare\blacksquare \\ \textbf{springst}\blacksquare\blacksquare\blacksquare \\ \blacksquare\blacksquare\textbf{brechen}\blacksquare \\ \blacksquare\blacksquare\blacksquare\blacksquare\blacksquare \\ \textbf{springe}\blacksquare\blacksquare\blacksquare \\ \blacksquare\blacksquare\blacksquare\textbf{breche} \\ \blacksquare\blacksquare\textbf{brichst}\blacksquare \\ \blacksquare\textbf{breche}\blacksquare\blacksquare \\ \hline \end{array}\right)$$

Tokens

$$\left.\begin{array}{|l|l|l|l|l|}\hline \text{infinitive} & \textbf{brechen} & & & \\ \hline \text{1st} & \text{breche?} & \text{brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \text{2nd} & \textbf{brichst} & \textbf{brecht} & \text{brichtest? brachst?} & \text{brichtet? bracht?} \\ \hline \text{3rd} & \text{bricht? brecht?} & \text{brichen? brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline & \text{singular} & \text{plural} & \text{singular} & \text{plural} \\ \hline & \multicolumn{2}{|c|}{\text{present}} & \multicolumn{2}{c|}{\text{past}} \\ \hline \end{array}\right)$$

Types

Overview

1. **String pairs**

$$p\left(\begin{array}{|c|c|}\hline \text{infinitive} & \text{brechen} \\ \hline \text{1st} & \text{breche?} \\ \hline \end{array}\right)$$

2. Multiple strings (paradigms)

$$p\left(\begin{array}{|c|c|c|c|c|}\hline \text{infinitive} & \text{brechen} & & & \\ \hline \text{1st} & \text{breche?} & \text{brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \text{2nd} & \text{brichst} & \text{brecht} & \text{brichtest? brachst?} & \text{brichtet? bracht?} \\ \hline \text{3rd} & \text{bricht? brecht?} & \text{brichen? brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \multicolumn{2}{|c|}{\text{singular}} & \text{plural} & \text{singular} & \text{plural} \\ \hline \multicolumn{2}{|c|}{\text{present}} & \multicolumn{3}{c|}{\text{past}} \\ \hline \end{array}\right)$$

3. Text and paradigms

$$p\left(\begin{array}{c} \text{German text} \\ \boxed{\begin{array}{l} \text{brichst} \\ \text{brecht} \\ \text{springst} \\ \text{brechen} \\ \text{springe} \\ \text{breche} \\ \text{brichst} \\ \text{breche} \end{array}} \\ \textbf{Tokens} \end{array}, \begin{array}{c} \begin{array}{|c|c|c|c|c|}\hline \text{infinitive} & \text{brechen} & & & \\ \hline \text{1st} & \text{breche?} & \text{brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \text{2nd} & \text{brichst} & \text{brecht} & \text{brichtest? brachst?} & \text{brichtet? bracht?} \\ \hline \text{3rd} & \text{bricht? brecht?} & \text{brichen? brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \multicolumn{2}{|c|}{\text{singular}} & \text{plural} & \text{singular} & \text{plural} \\ \hline \multicolumn{2}{|c|}{\text{present}} & \multicolumn{3}{c|}{\text{past}} \\ \hline \end{array} \\ \textbf{Types} \end{array}\right)$$

1 String Pairs

String pair problems are common in NLP:

Morphology:

infinitive	brechen
1st	breche?

1 String Pairs

String pair problems are common in NLP:

Morphology:

infinitive	brechen
1st	breche?

Transliteration

john hardt - yue han ha te

patrick johnson - pa te li ke yue han xun

frederick william mulley - fu lei de li ke wei lian ma li

Pronunciation

Sternanisöl - /ˈʃtɛrnʔaniːsʔøːl/

loophole - /ˈluːpˌhəʊl/

Spelling correction

Honululu - Honolulu

braek- break

1 String Pairs

- We want to build a **probability model** over string pairs.
- Such a model can produce k-best output, can be **plugged** into bigger models later, etc.
- We would like to make use of **flexible features**, be able to look at **linguistic properties** of the strings,
- and **train** the parameters from data.

1 String Pairs

$$Pr(s_1,s_2) = \frac{1}{Z} F(s_1,s_2)$$

$$Pr(s_2 \mid s_1) = \frac{1}{Z} F(s_1,s_2)$$

- Function F evaluates how well the two strings go together.
- It looks at properties ("features") of the string pair and assigns some score.

1 String Pairs

s_1= breaking
s_2= broke

1 String Pairs

s_1= #breaking#
s_2= #broke#

1 String Pairs

#breaking#
#broke#

1 String Pairs

#breaking#
#br∈oke∈∈#

1 String Pairs

#breaking#
#br∈oke∈∈#

#breaking#
#bro∈ke∈∈#

#brea∈king#
#br∈∈oke∈∈#

#break∈ing#
#bro∈ke∈∈∈#

…

1 String Pairs

s_1= #breaking#
s_2= #broke#

1 String Pairs

s_1= #breaking#
s_2= #broke#

$F(s_1,s_2)$ =

1 String Pairs

s_1= #breaking#
s_2= #broke#

$F(s_1,s_2)$ =

1 String Pairs

s_1= #breaking#
s_2= #broke#

$F(s_1,s_2)$ =

$$\exp \sum_i \theta_i f_i\left(\begin{array}{c} \text{\#breaking\#} \\ \text{\#br}\epsilon\text{oke}\epsilon\epsilon\text{\#} \end{array} \right)$$

1 String Pairs

s_1= #breaking#
s_2= #broke#

$F(s_1, s_2) =$

$$\left(\exp \sum_i \theta_i f_i(\begin{matrix} \texttt{\#breaking\#} \\ \texttt{\#br}\epsilon\texttt{oke}\epsilon\epsilon\texttt{\#} \end{matrix}) + \exp \sum_i \theta_i f_i(\begin{matrix} \texttt{\#breaking\#} \\ \texttt{\#bro}\epsilon\texttt{ke}\epsilon\epsilon\texttt{\#} \end{matrix}) \right)$$

1 String Pairs

s_1= #breaking#
s_2= #broke#

$F(s_1, s_2) =$

$$\exp \sum_i \theta_i f_i(\begin{matrix}\#breaking\# \\ \#br\epsilon oke\epsilon\epsilon\#\end{matrix})$$
$$+ \exp \sum_i \theta_i f_i(\begin{matrix}\#breaking\# \\ \#bro\epsilon ke\epsilon\epsilon\#\end{matrix})$$
$$+ \exp \sum_i \theta_i f_i(\begin{matrix}\#brea\epsilon king\# \\ \#br\epsilon\epsilon oke\epsilon\epsilon\#\end{matrix})$$

1 String Pairs

s_1= #breaking#
s_2= #broke#

$$F(s_1,s_2) =$$

$$\left(\begin{array}{l}
\exp\sum_i \theta_i f_i(\begin{array}{l}\texttt{\#breaking\#}\\ \texttt{\#br}\epsilon\texttt{oke}\epsilon\epsilon\texttt{\#}\end{array}) \\
+\exp\sum_i \theta_i f_i(\begin{array}{l}\texttt{\#breaking\#}\\ \texttt{\#bro}\epsilon\texttt{ke}\epsilon\epsilon\texttt{\#}\end{array}) \\
+\exp\sum_i \theta_i f_i(\begin{array}{l}\texttt{\#brea}\epsilon\texttt{king\#}\\ \texttt{\#br}\epsilon\epsilon\texttt{oke}\epsilon\epsilon\texttt{\#}\end{array}) \\
+\exp\sum_i \theta_i f_i(\begin{array}{l}\texttt{\#break}\epsilon\texttt{ing\#}\\ \texttt{\#bro}\epsilon\texttt{ke}\epsilon\epsilon\epsilon\texttt{\#}\end{array})
\end{array}\right)$$

1 String Pairs

s_1= #breaking#
s_2= #broke#

$F(s_1, s_2) =$

$$\left(\begin{array}{l}
\exp \sum_i \theta_i f_i(\begin{array}{l}\texttt{\#breaking\#}\\ \texttt{\#br}\epsilon\texttt{oke}\epsilon\epsilon\texttt{\#}\end{array}) \\
+ \exp \sum_i \theta_i f_i(\begin{array}{l}\texttt{\#breaking\#}\\ \texttt{\#bro}\epsilon\texttt{ke}\epsilon\epsilon\texttt{\#}\end{array}) \\
+ \exp \sum_i \theta_i f_i(\begin{array}{l}\texttt{\#brea}\epsilon\texttt{king\#}\\ \texttt{\#br}\epsilon\epsilon\texttt{oke}\epsilon\epsilon\texttt{\#}\end{array}) \\
+ \exp \sum_i \theta_i f_i(\begin{array}{l}\texttt{\#break}\epsilon\texttt{ing\#}\\ \texttt{\#bro}\epsilon\texttt{ke}\epsilon\epsilon\epsilon\texttt{\#}\end{array}) \\
+ \ldots
\end{array}\right)$$

1

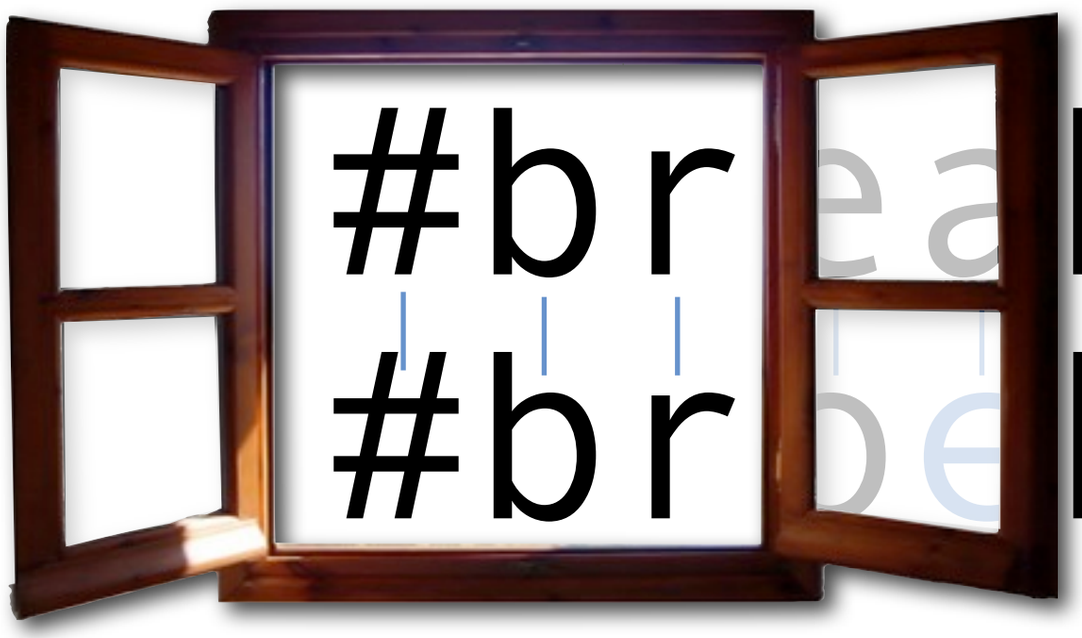
String Pairs

$$\exp \sum_i \theta_i f_i\left(\begin{array}{c}\texttt{\#breakєing\#}\\ \texttt{\#broєkeєєє\#}\end{array}\right)$$

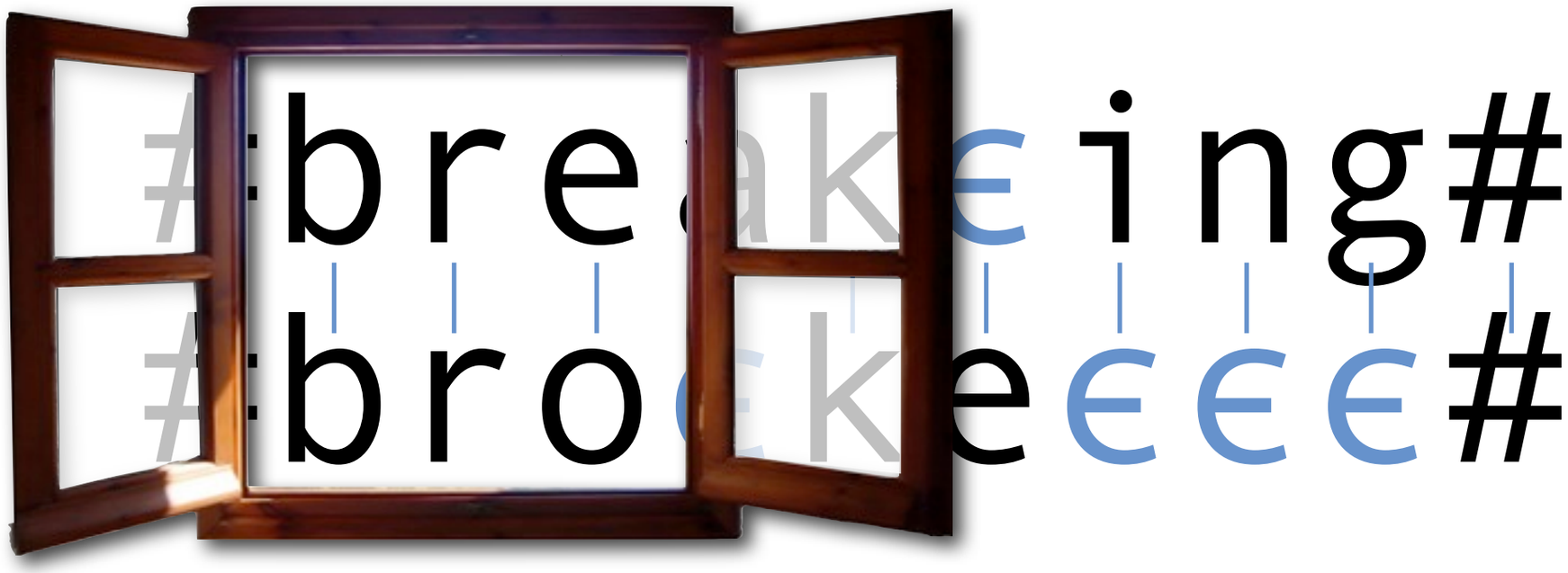
#breakϵing#

#broϵkeϵϵϵ#

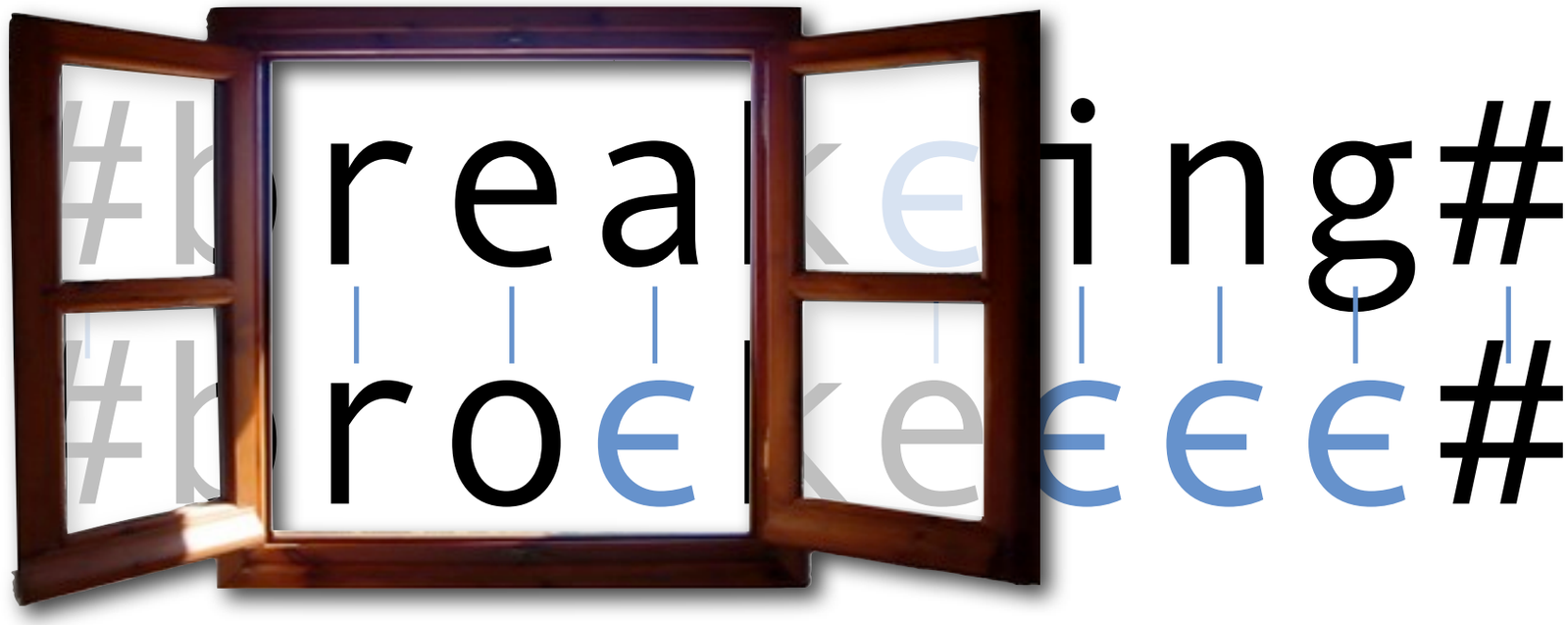
$$\exp \sum_i \theta_i f_i \left(\begin{array}{c} \texttt{\#break}\epsilon\texttt{ing\#} \\ \texttt{\#bro}\epsilon\texttt{ke}\epsilon\epsilon\epsilon\texttt{\#} \end{array} \right)$$



$$\exp \sum_i \theta_i f_i(\text{\#breaking\#}, \text{\#broeke€€€\#})$$

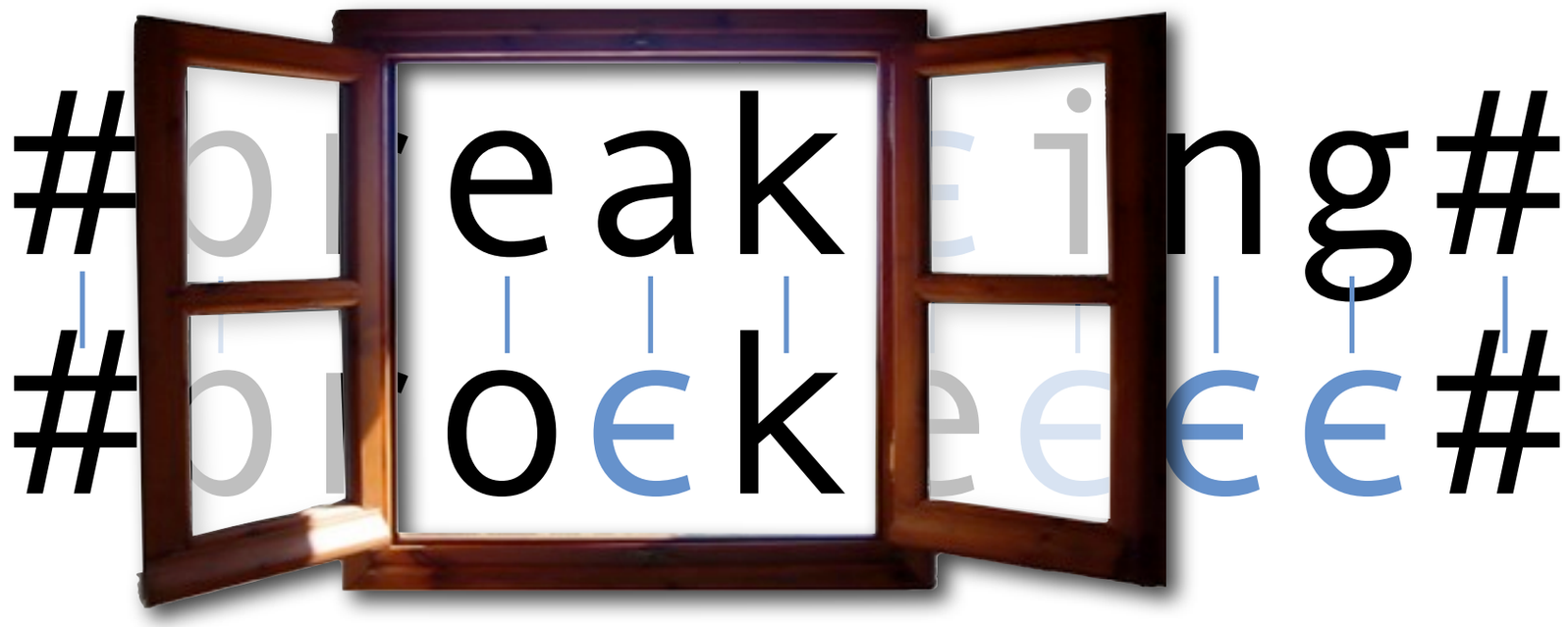


$$\exp \sum_i \theta_i f_i(\begin{matrix}\texttt{\#breakєing\#} \\ \texttt{\#broєkeєєє\#}\end{matrix})$$

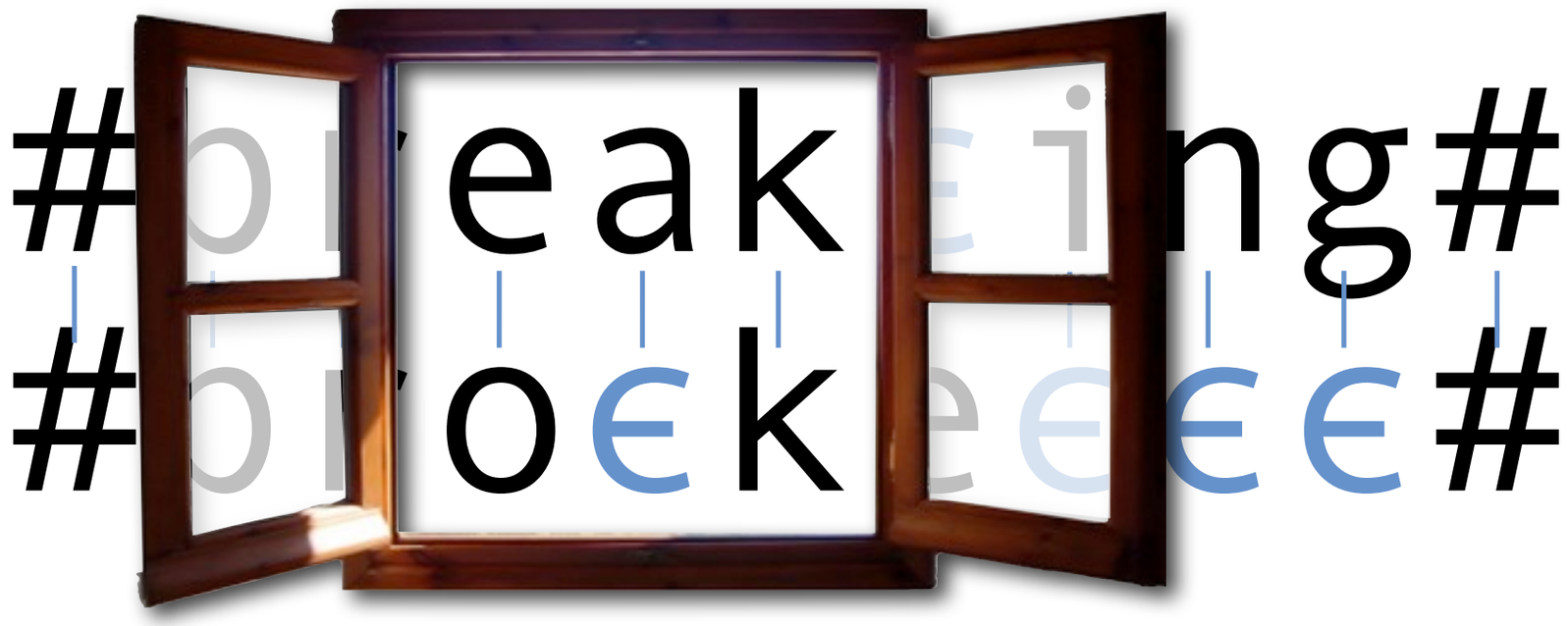


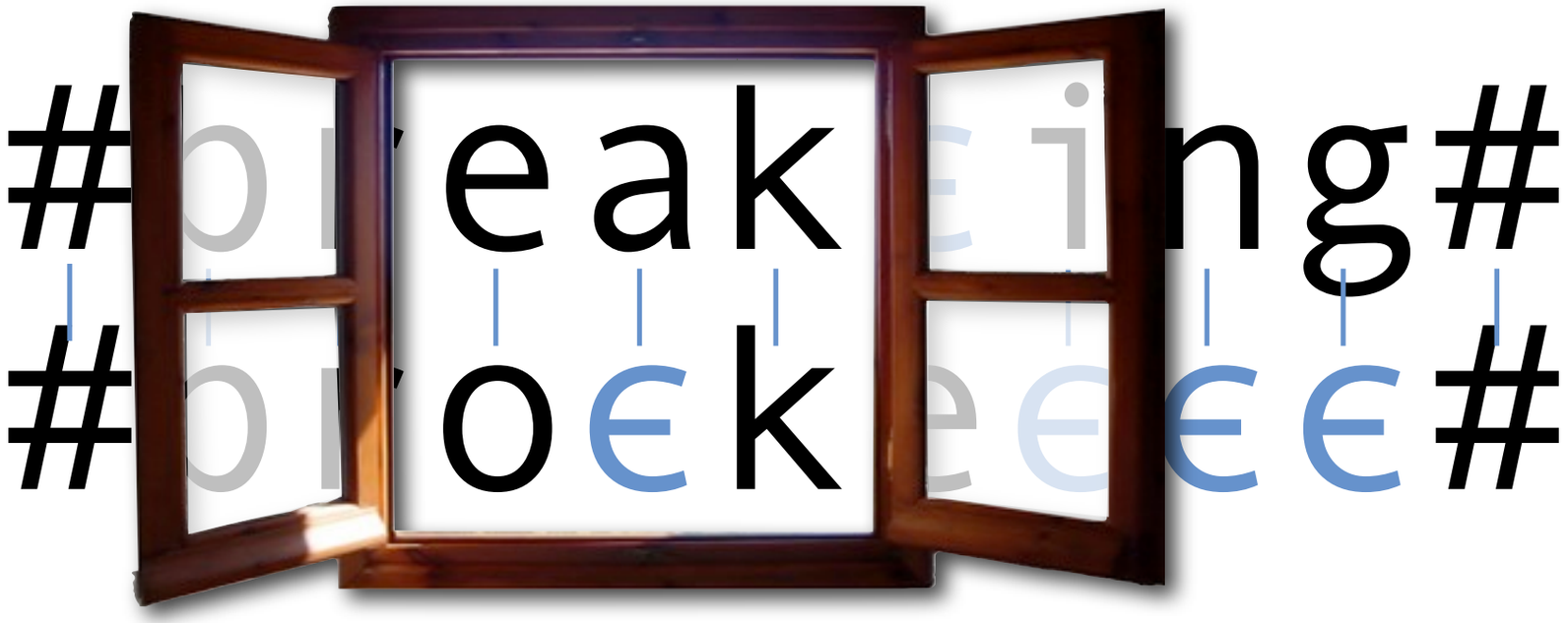
$$\exp \sum_i \theta_i f_i(\begin{matrix}\texttt{\#breakεing\#} \\ \texttt{\#broεkeεεε\#}\end{matrix})$$

$$\exp \sum_i \theta_i f_i(\begin{matrix}\texttt{\#breakεing\#} \\ \texttt{\#broεkeεεε\#}\end{matrix})$$

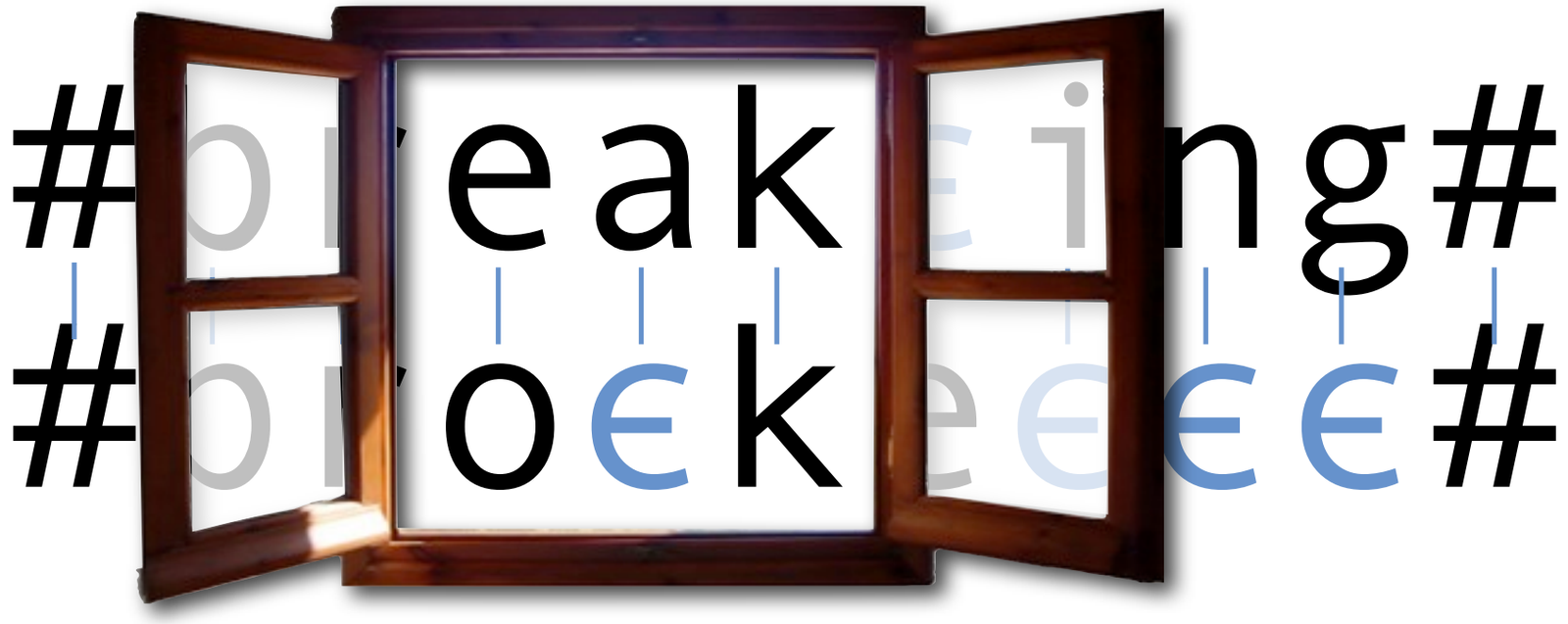


$$\exp \sum_i \theta_i f_i\Big(\begin{array}{l} \texttt{\#break}\epsilon\texttt{ing\#} \\ \texttt{\#bro}\epsilon\texttt{ke}\epsilon\epsilon\epsilon\texttt{\#} \end{array} \Big)$$

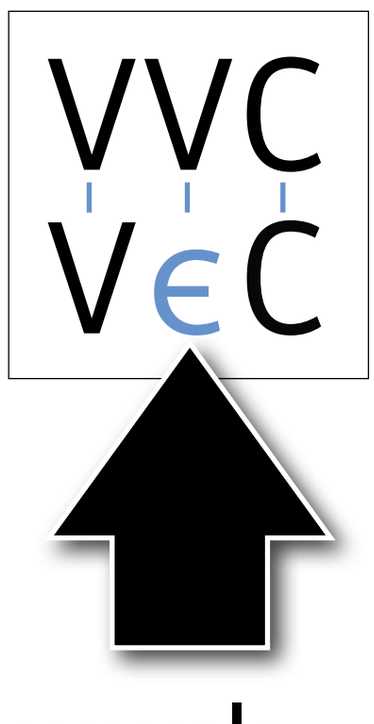

#breaking#
#broeken#



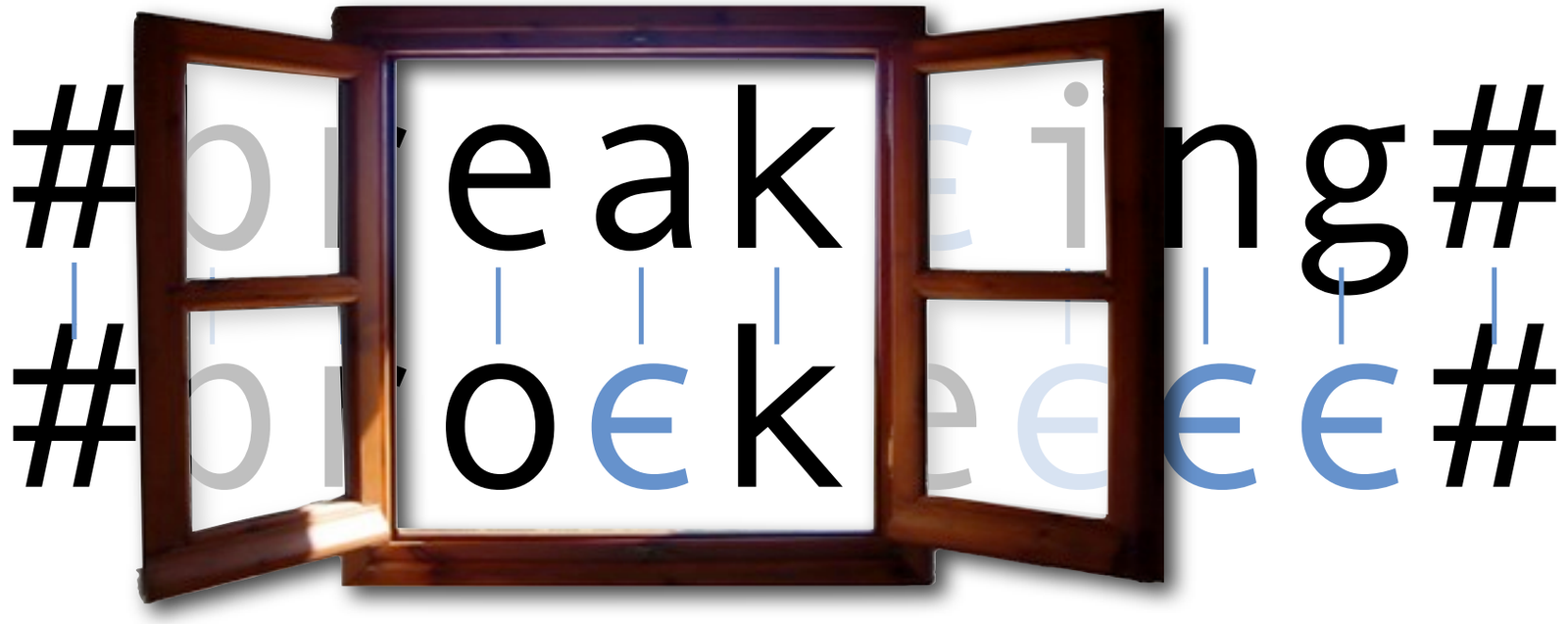
full
window

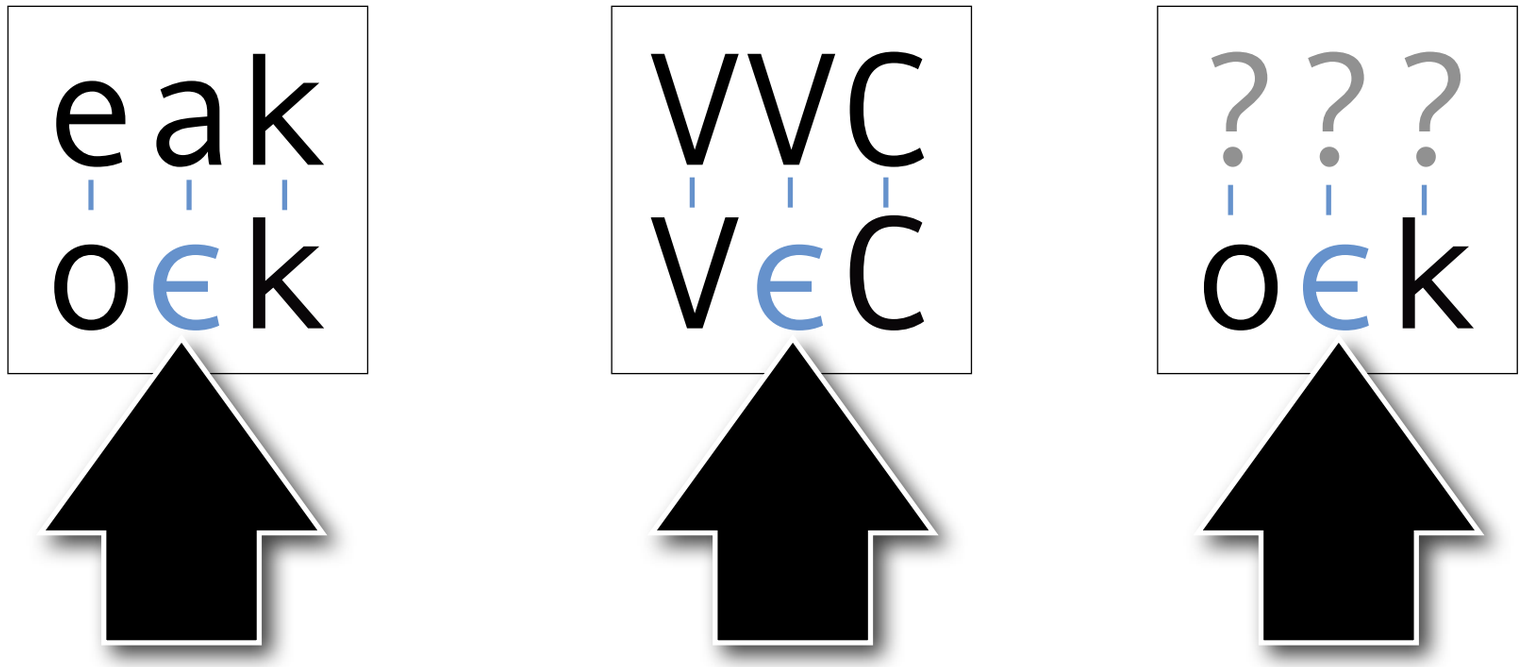


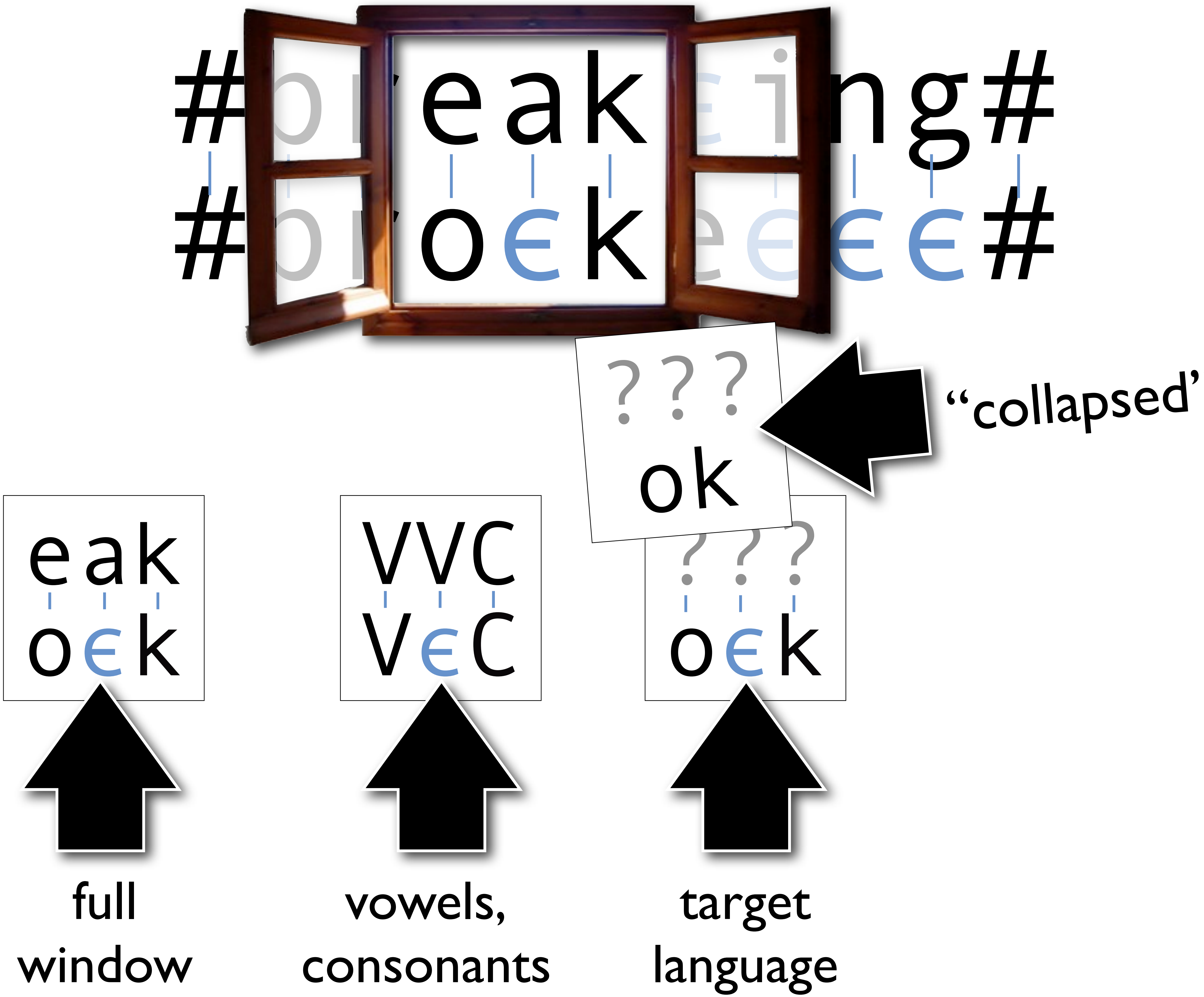
full window



vowels, consonants


#breaking#
#broɛkɛɛɛɛ#


eak
oɛk
full window
VVC
VɛC
vowels, consonants
???
oɛk
target language



eak
oεk
???
ok
“collapsed”
eak
oεk
VVC
VεC
???
oεk
full window
vowels, consonants
target language

#b r eak i ng#
#b r oϵk eϵϵϵ#

???
ok
???

"collapsed"

eak
oϵk

full window

VVC
VϵC

vowels, consonants

???
oϵk

target language

subst
del
ident

subst, del, ins, ident

Also add versions of these features that are backed off to bigrams!

k i n g #

k e ϵ ϵ ϵ #

? ? ?
? k

"collapsed"

? a k
? ϵ k

full window

? V C
? ϵ C

vowels, consonants

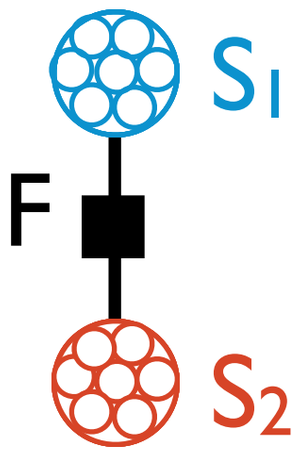
? ? ?
? ϵ k

target language

? : ?
? : del
ident

subst, del, ins, ident

1 String Pairs



- To compute such feature-based scores for two string variables S_1 and S_2, we construct a **weighted finite-state transducer F**
- It can assign a **score** to any string pair s_1, s_2

$$\Pr(s_1, s_2) = \frac{1}{Z} F(s_1, s_2)$$

Background: Finite-state machines

What is a **finite-state *acceptor* (FSA)**?

An automaton with a finite number of states and arcs.

Can be used to assign a score to any **string**.

What is a **finite-state *transducer* (FST)**?

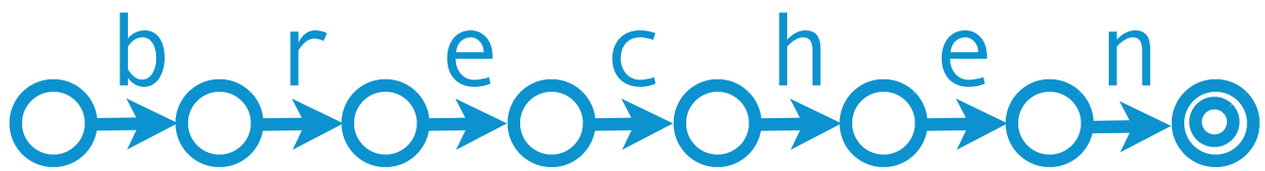
Same as FSA, but used to assign score to any **string pair**

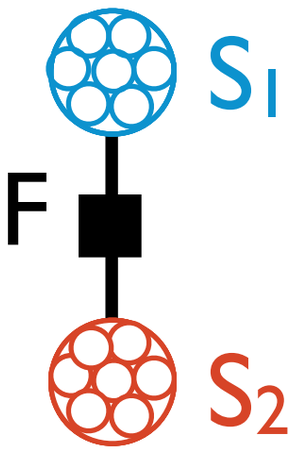
(e.g. evaluating how well they go together).

1 String Pairs

- Specific kind of **grammar** that describes and scores one or more strings
- **Closure** properties under many useful operations (we will use composition, intersection, projection)
- **Useful** for many tasks in natural language processing

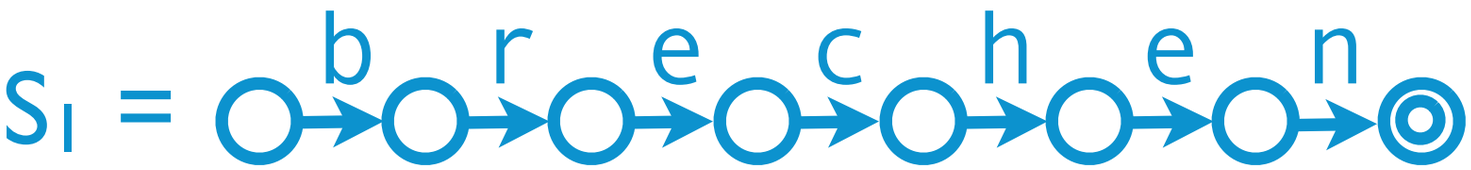
1 String Pairs

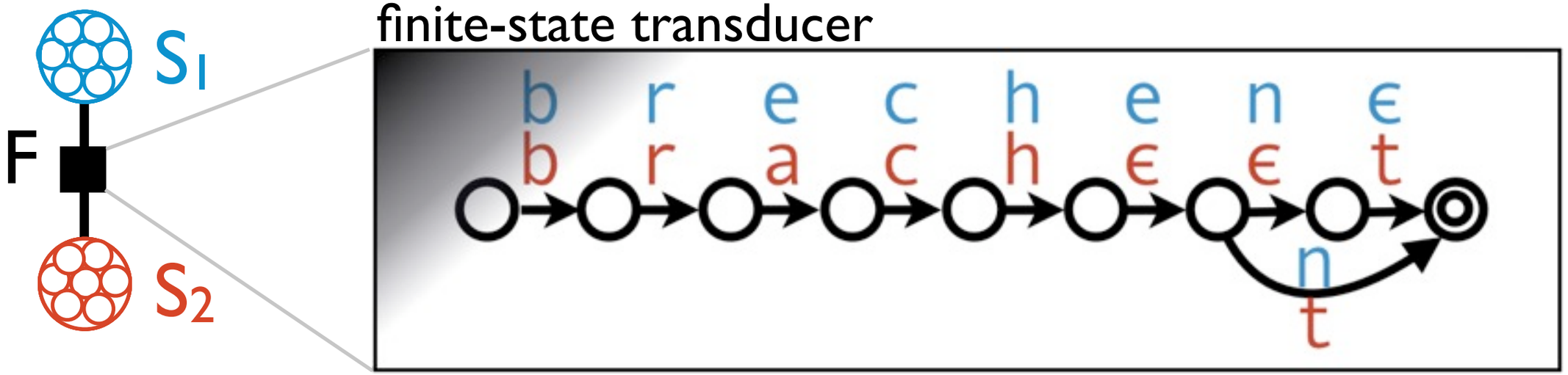
S_1 = 

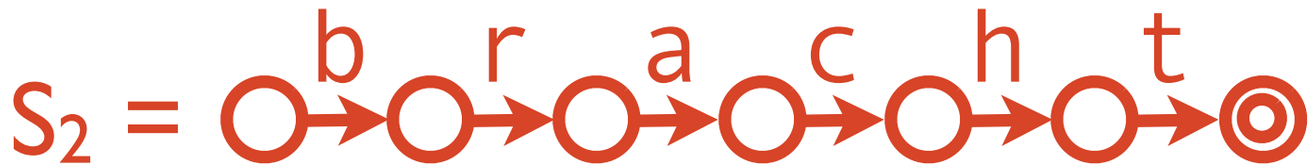


S_2 = 

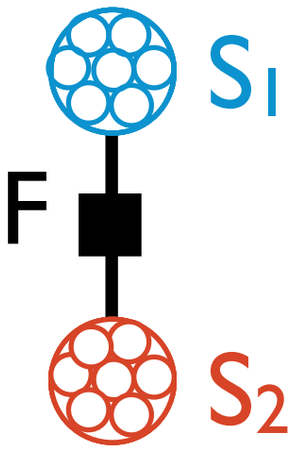
1 String Pairs



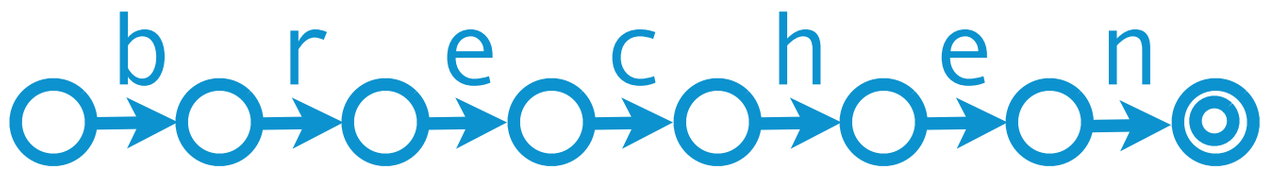




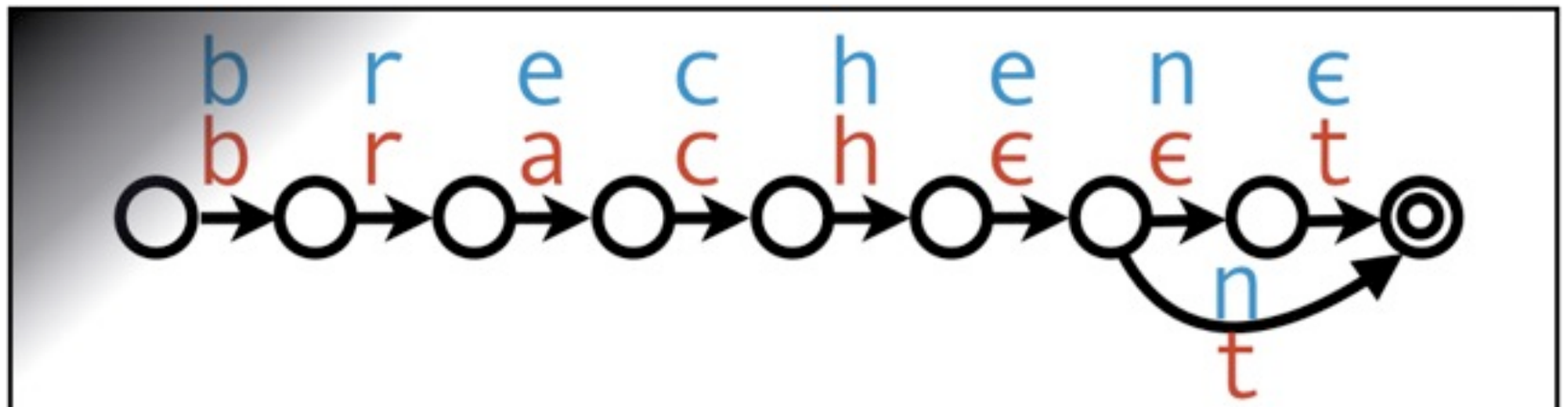
1 String Pairs



S_1 =

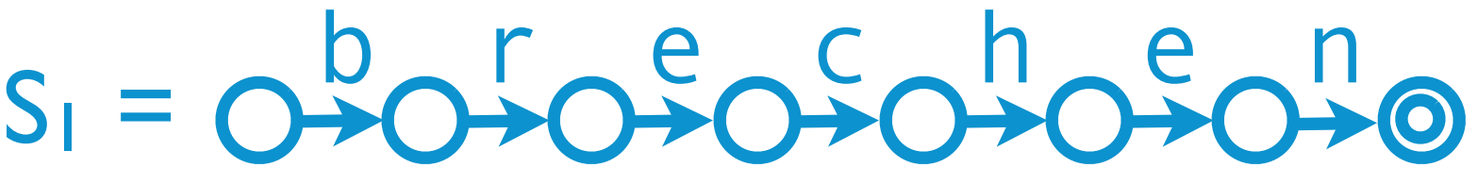


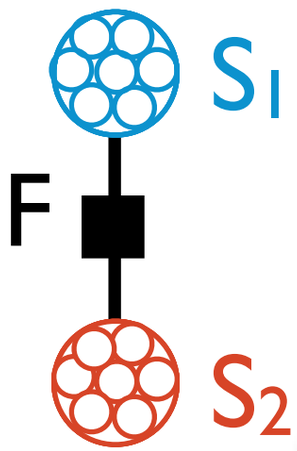
finite-state transducer



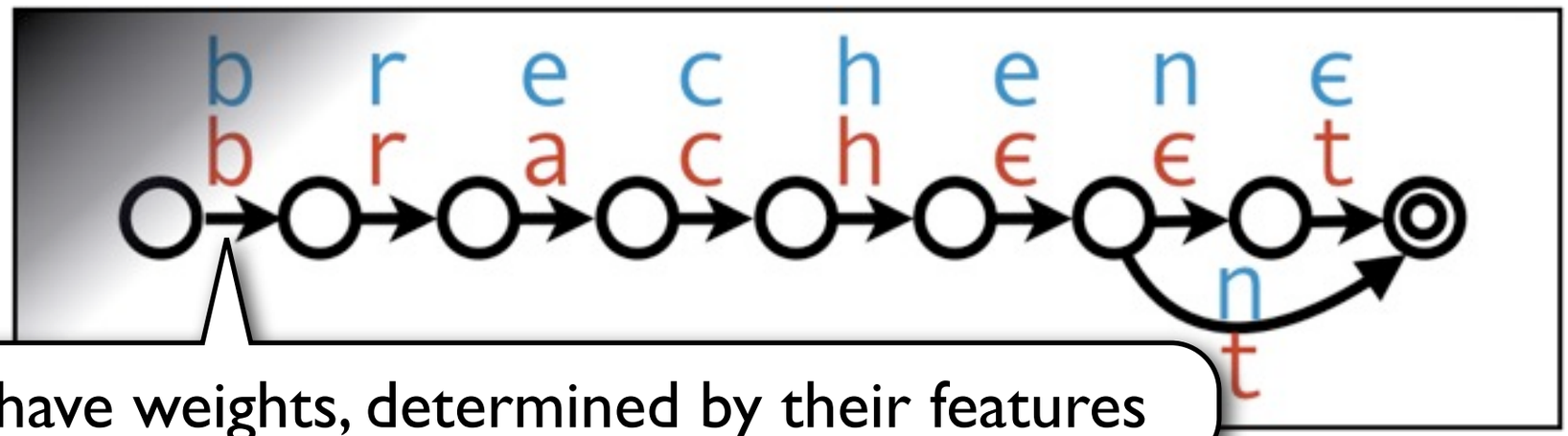
S_2 =

1 String Pairs



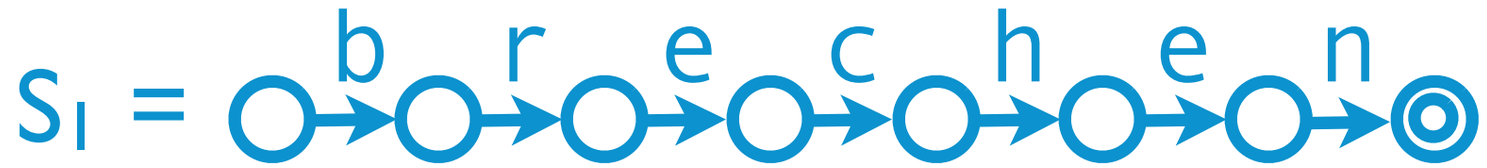


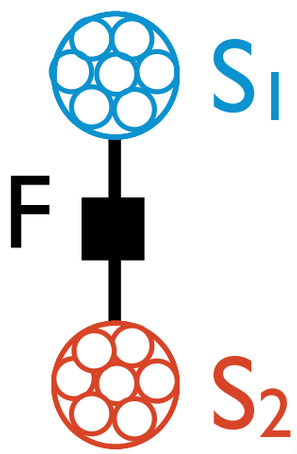
finite-state transducer



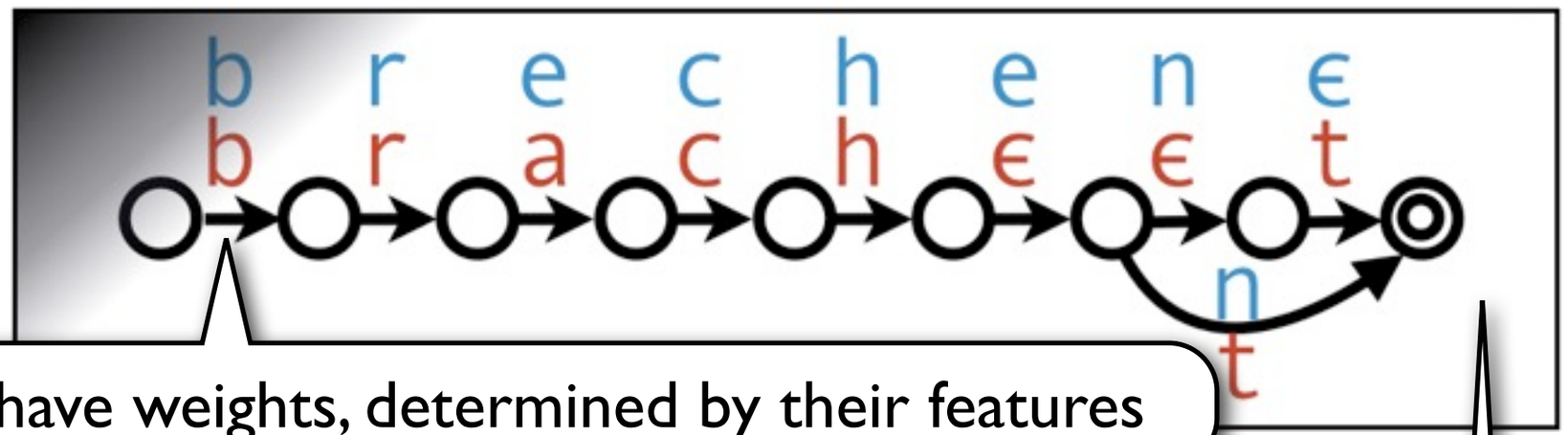
arc have weights, determined by their features

1 String Pairs





finite-state transducer



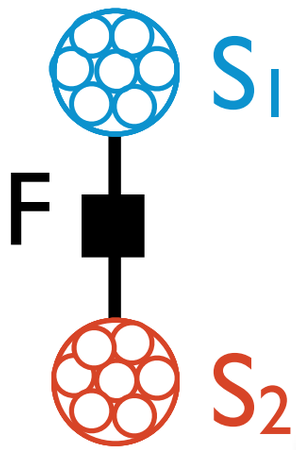
arc have weights, determined by their features

Transducer F computes score by looking at **all** alignments

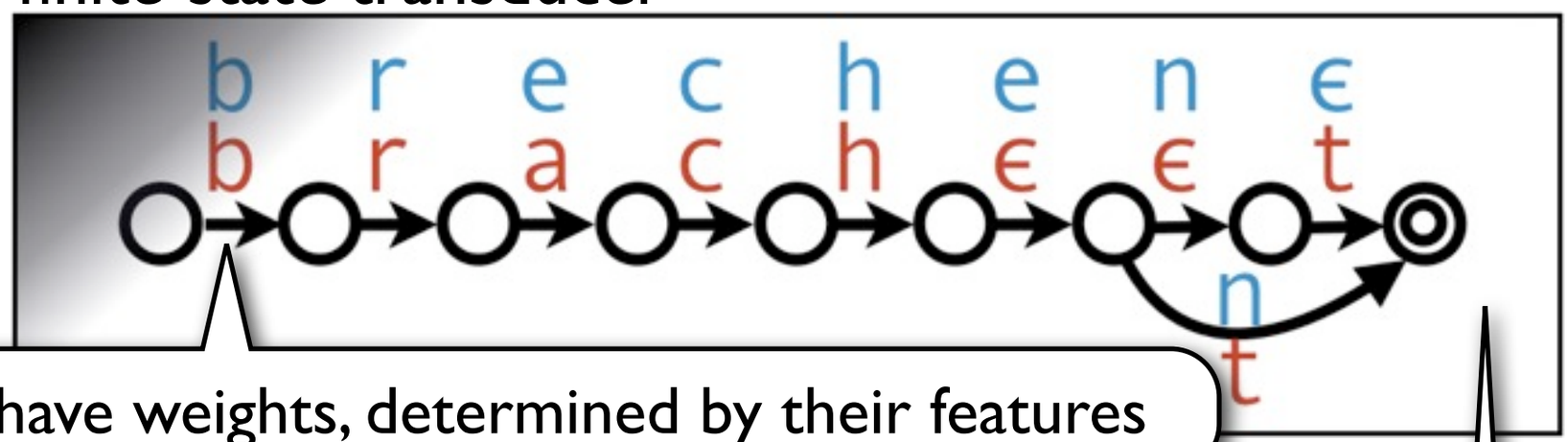
1 String Pairs

Sum over all paths in the finite-state transducer

S_1 = b r e c h e n



finite-state transducer



=13.26

arc have weights, determined by their features

S_2 = b r a c h t

Transducer F computes score by looking at **all** alignments

1 String Pairs

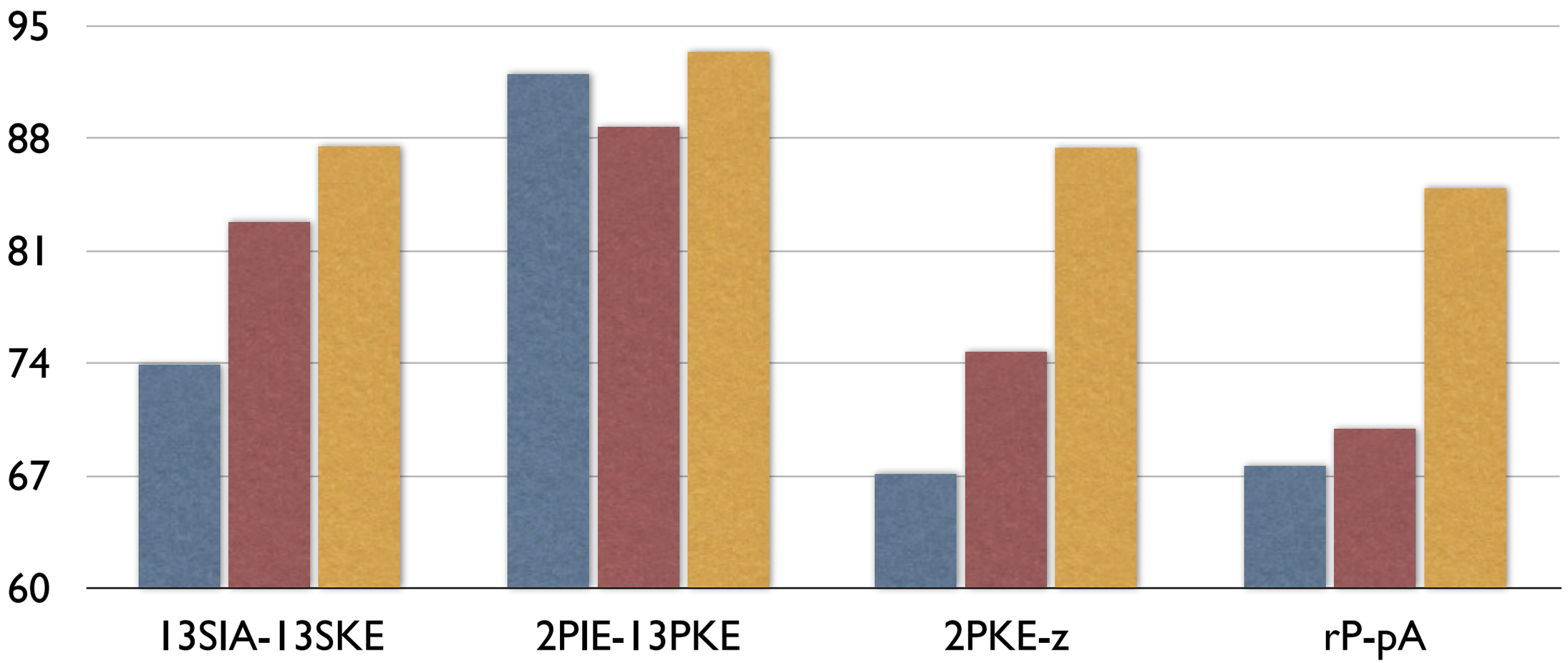
- The alignment between the string pair is a latent variable.
- We add more latent variables to the model:
  - Change regions
  - Conjugations classes

For details, see my thesis, and Dreyer, Smith & Eisner, 2008

1

String Pairs

Inflection (on German verbs)



(baseline)

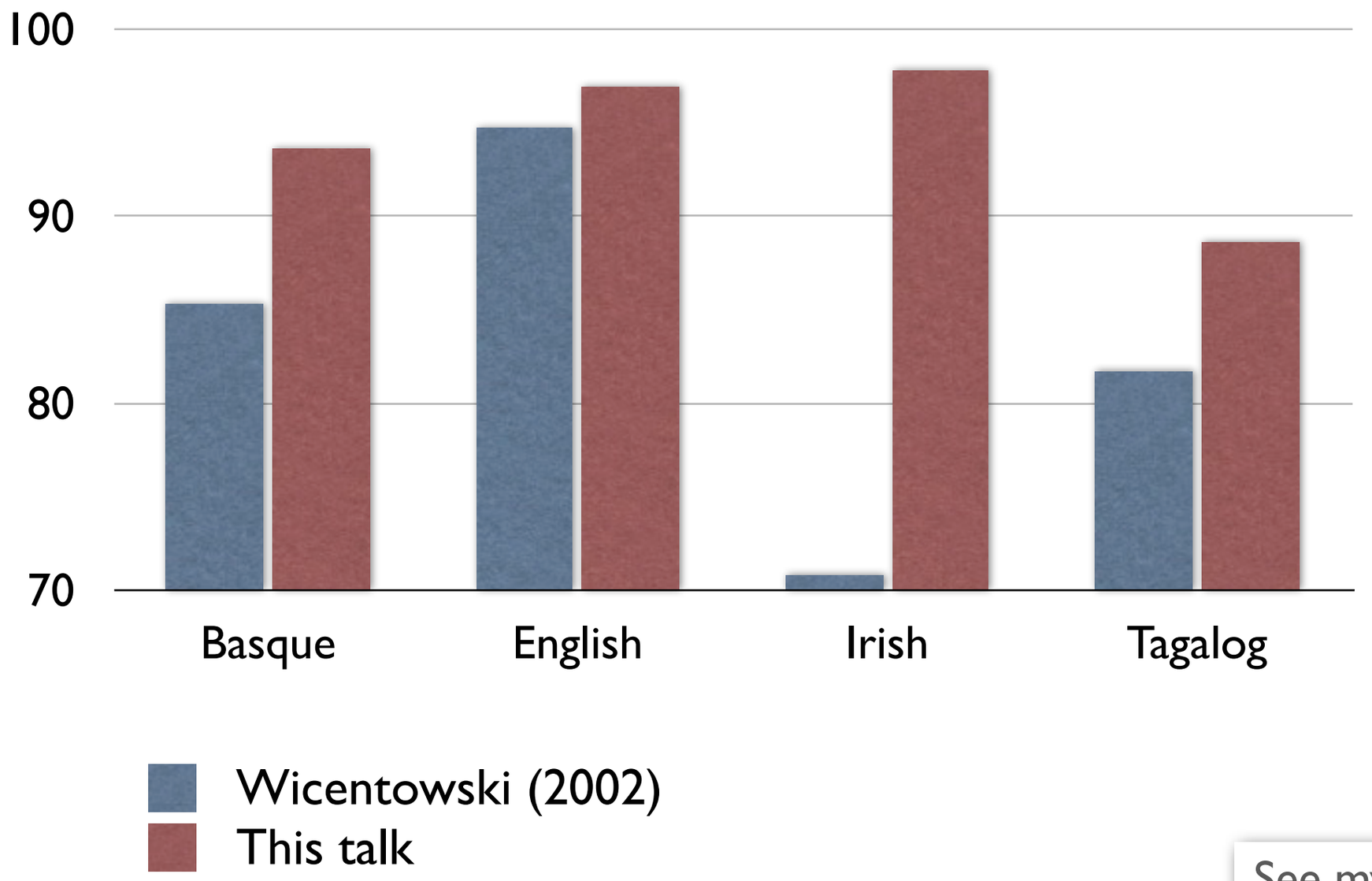
(this talk)

See my thesis, and Dreyer, Smith & Eisner, 2008

1

String Pairs

Lemmatization

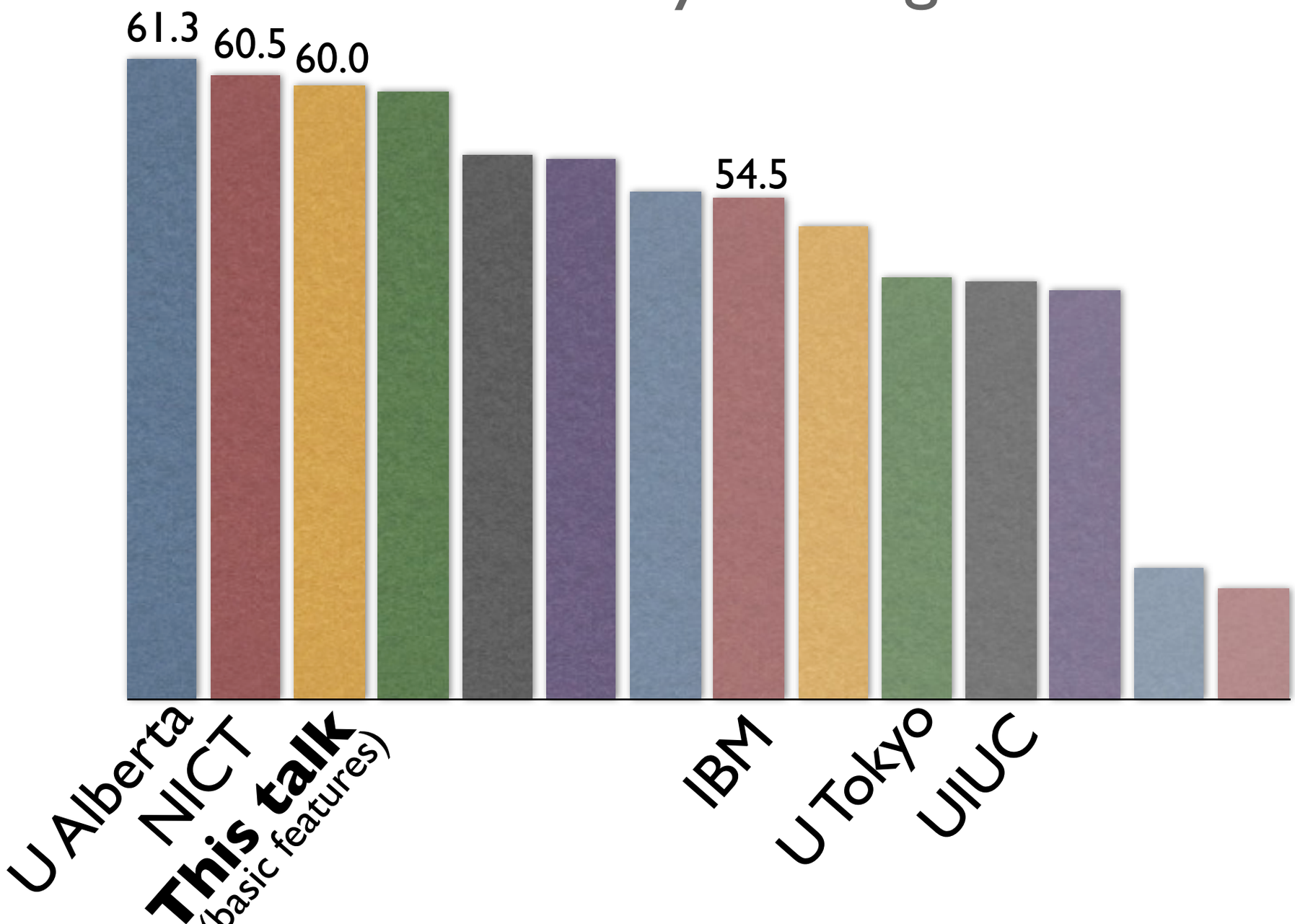


See my thesis, and Dreyer, Smith & Eisner, 2008

1 String Pairs

Transliteration competition, NEWS 2009

Accuracy on English-to-Russian



1 Conclusions / Contributions

- Presented a novel, well-defined **probability model** over string pairs (or single strings)
- **General** enough to model many string-to-string problems in NLP (and neighboring disciplines)
- Achieved high-scoring results in **different tasks** (inflection, lemmatization, transliteration) in **multiple languages** (*German, Basque, English, Irish, Tagalog, Russian)*

1 Conclusions / Contributions

- **Linguistic properties** and soft constraints can be expressed and learned *(prefer certain vowel/consonant sequences, prefer identities, ...)*
- **Arbitrary-length output** is handled elegantly (eliminates need for limiting structure insertion)
- Much information does not need to be annotated; it is inferred as **hidden variables** (alignments, conjugation classes, regions)

Overview

1 String pairs

$p($

infinitive	brechen
1st	breche?

$)$

2 **Multiple strings (paradigms)**

$p($

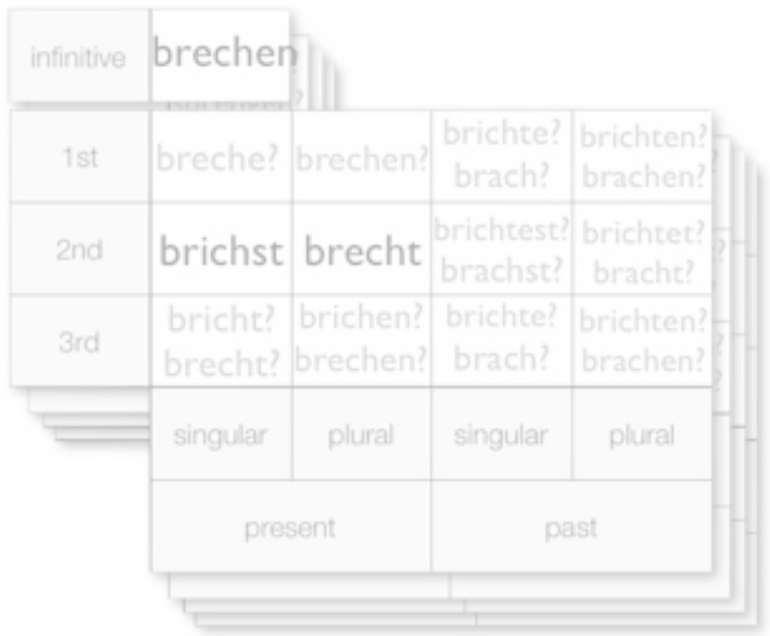
infinitive	brechen			
1st	breche?	brechen?	brichte? brach?	brichten? brachen?
2nd	brichst	brecht	brichtest? brachst?	brichtet? bracht?
3rd	bricht? brecht?	brichen? brechen?	brichte? brach?	brichten? brachen?
	singular	plural	singular	plural
	present		past	

$)$

3 Text and paradigms

$p($

Tokens



Types

$)$

2 Multiple Strings

infinitive	brechen
1st	breche?

- We've seen how to model 2 strings, using feature-based finite-state machines
- But we have bigger goals ...

2 Multiple Strings

infinitive	brechen
1st	breche?

2 Example applications

Inflectional paradigms

infinitive	brechen			
1st	breche?	brechen?	brichte? brach?	brichten? brachen?
2nd	brichst	brecht	brichtest? brachst?	brichtet? bracht?
3rd	bricht? brecht?	brichen? brechen?	brichte? brach?	brichten? brachen?
	singular	plural	singular	plural
	present		past	

2 Example applications

Inflectional paradigms

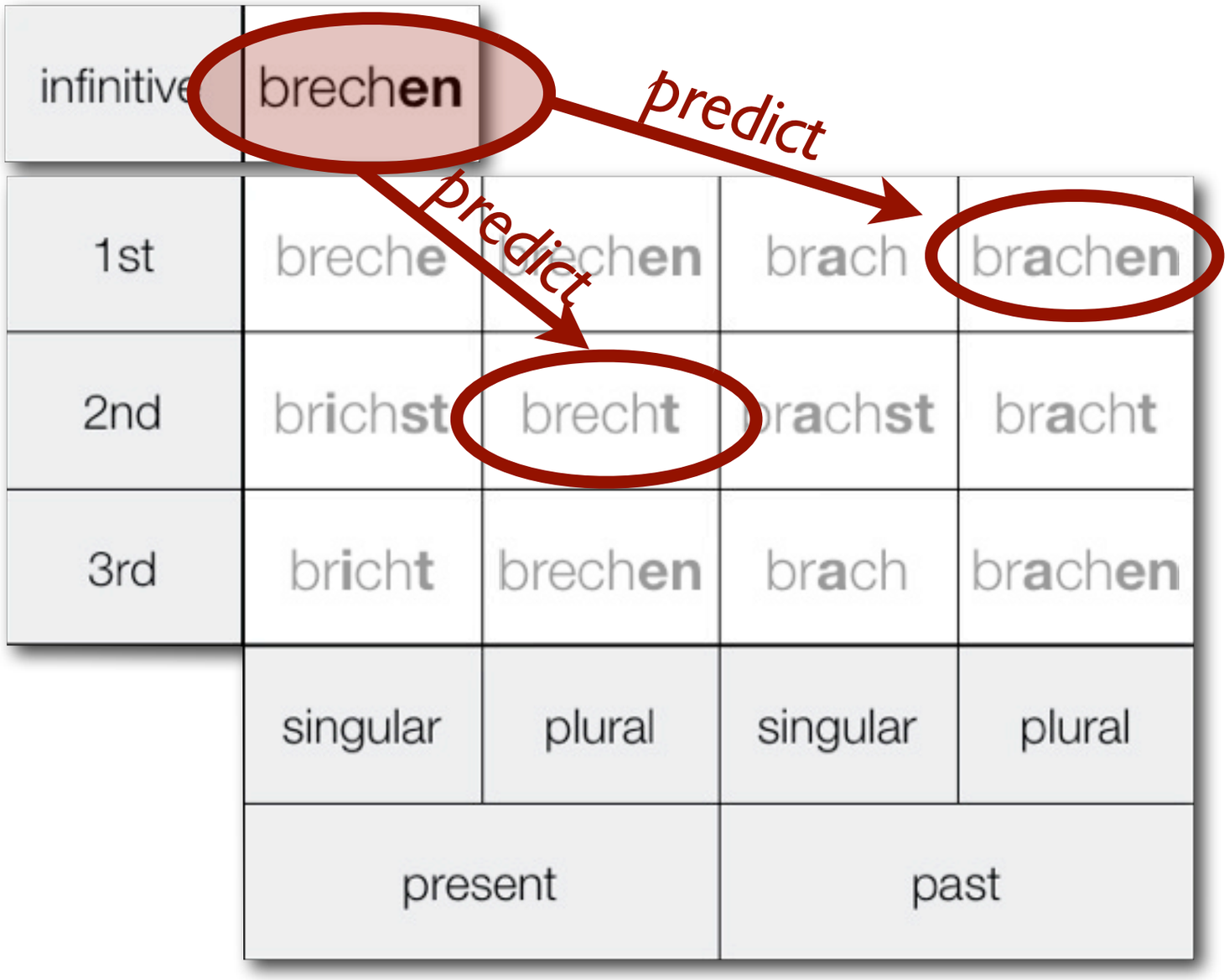
2 Example applications

Inflectional paradigms

infinitive	brech**en**			
1st	brech**e** ?	brech**en** ?	brach ?	brach**en** ?
2nd	brich**st** ?	brech**t** ?	brach**st** ?	brach**t** ?
3rd	brich**t** ?	brech**en** ?	brach ?	brach**en** ?
	singular	plural	singular	plural
	present		past	

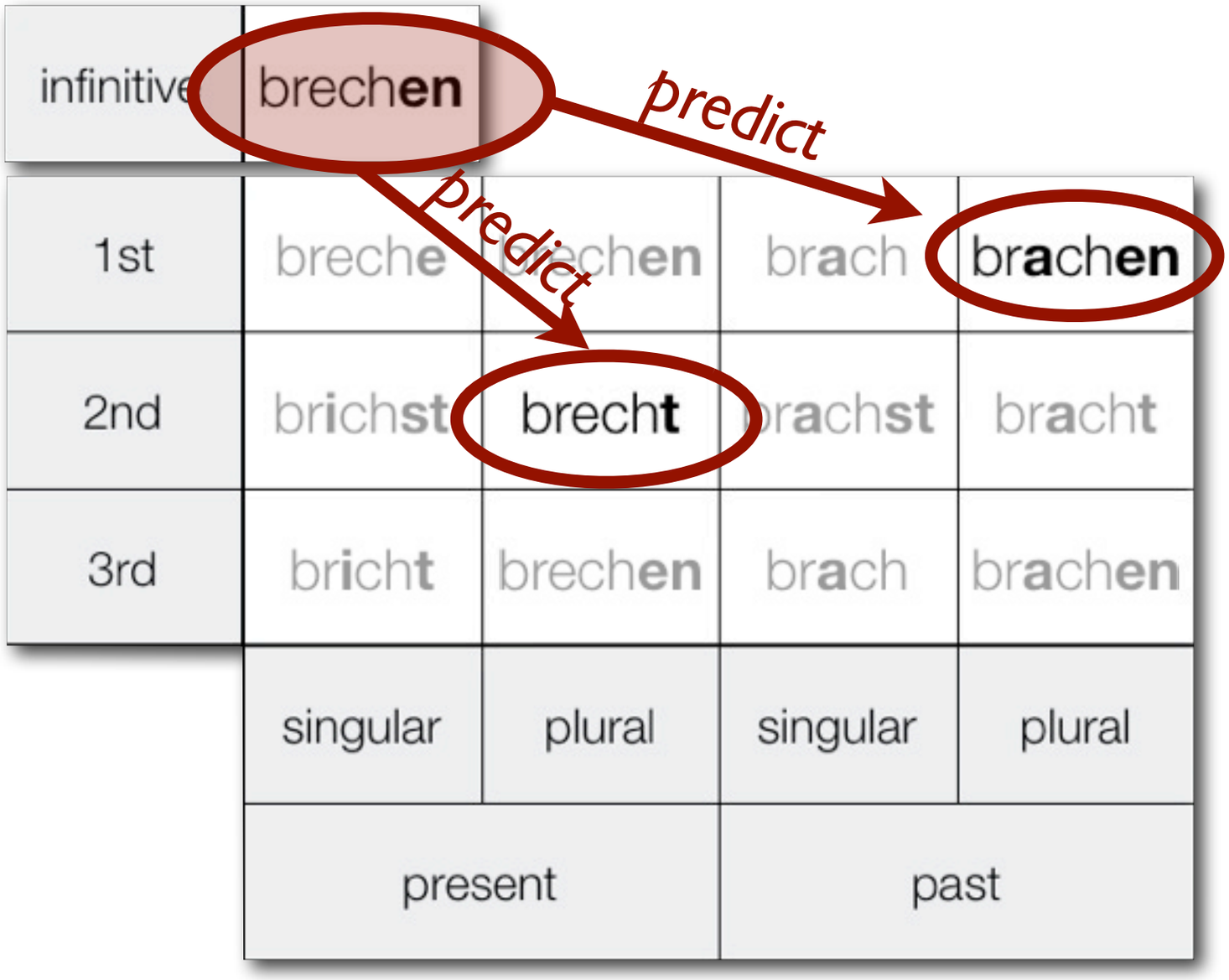
2 Example applications

Inflectional paradigms



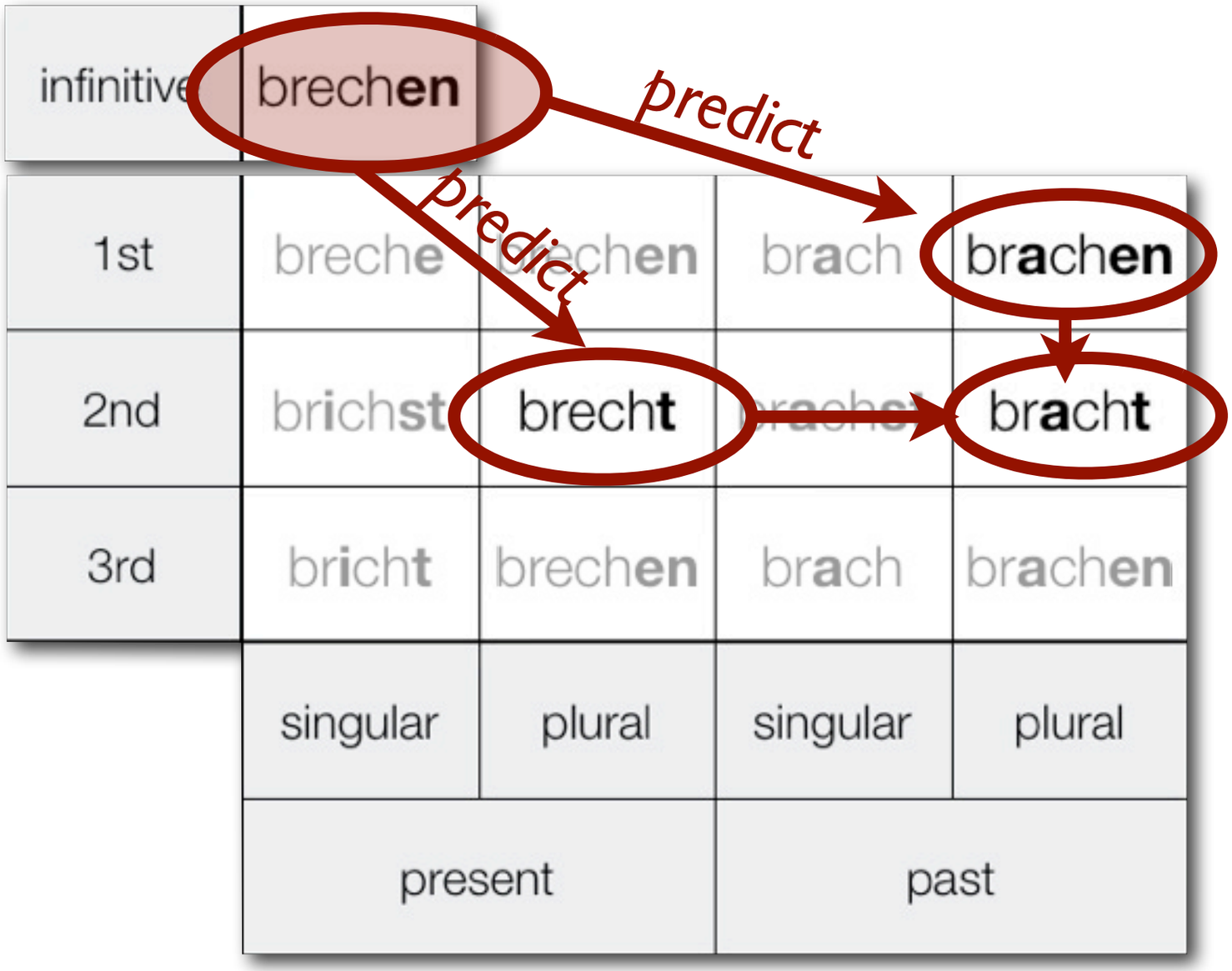
2 Example applications

Inflectional paradigms



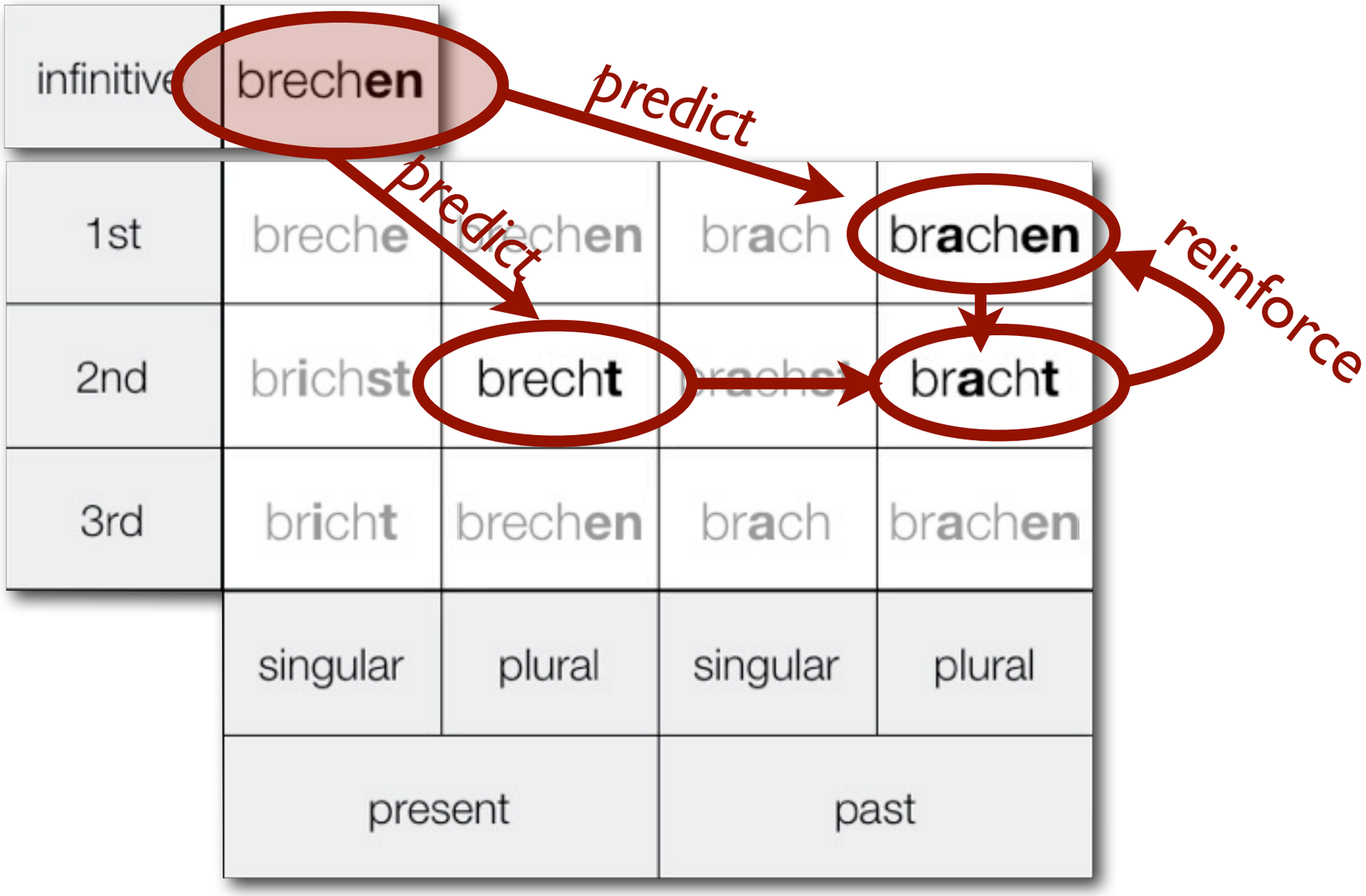
2 Example applications

Inflectional paradigms



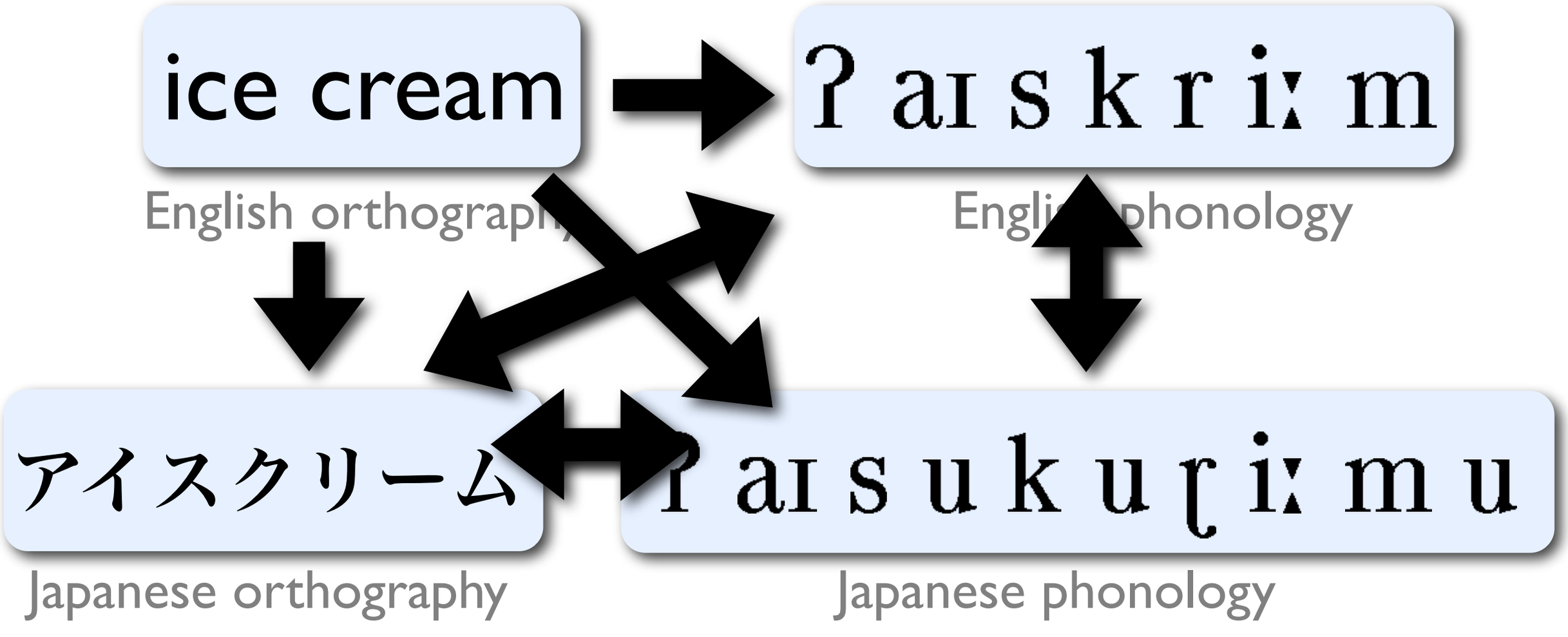
2 Example applications

Inflectional paradigms



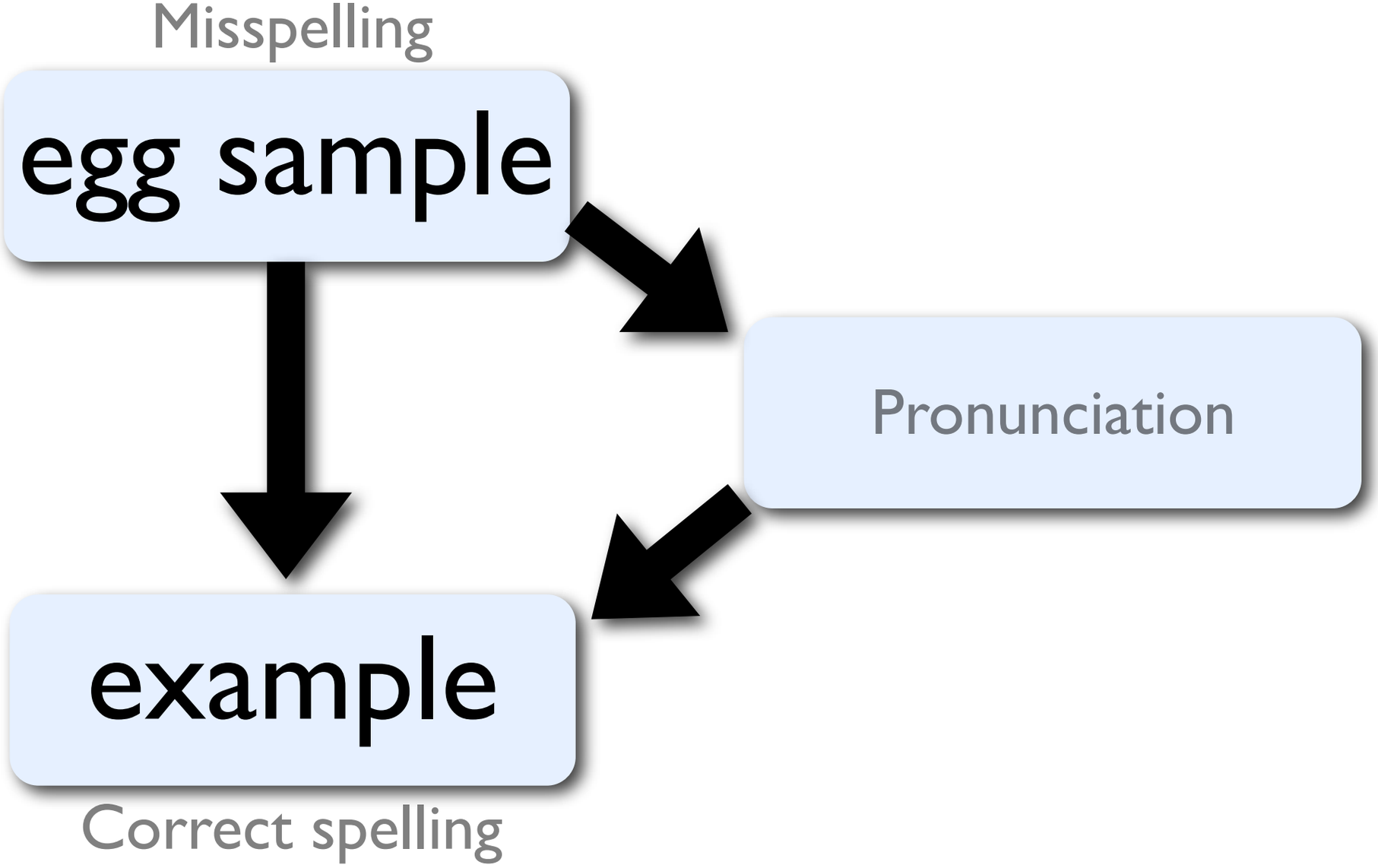
② Example applications

Transliteration (using phonology)



2 Example applications

Spelling correction



② Example applications

... and all other tasks where word forms and representations interact:

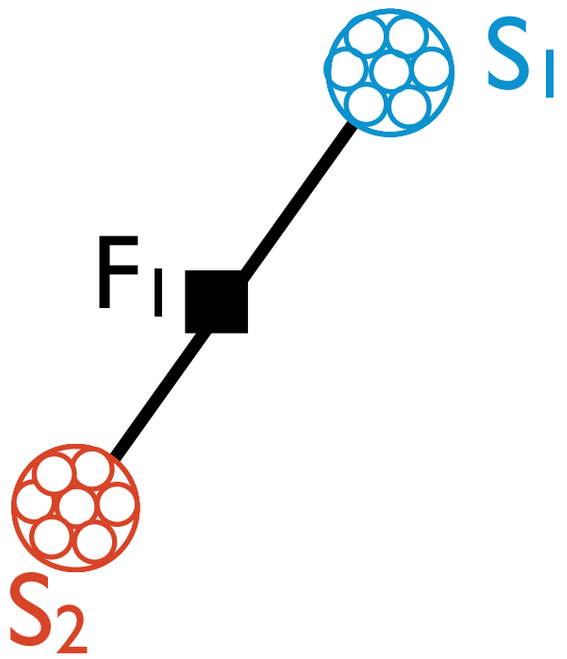
- Cognate modeling
- Multiple-string alignment
- System combination

2 Multiple Strings

- Let's build a general **probability model over multiple strings**
- It extends the string-pair model we saw in the last part.
- We will later be able to use it to learn how to inflect verbs.

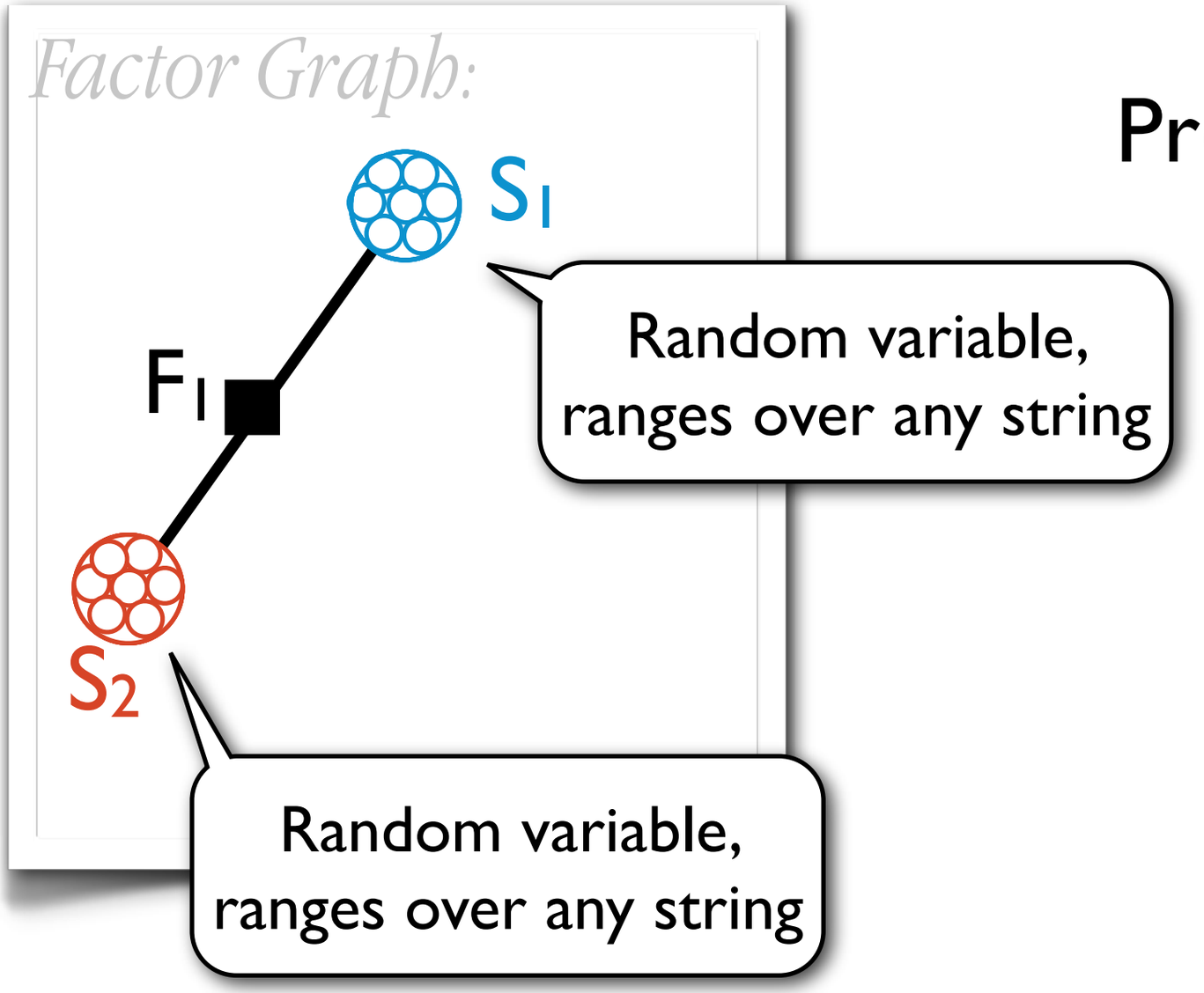
2 Model. *Factor graph examples*

Factor Graph:



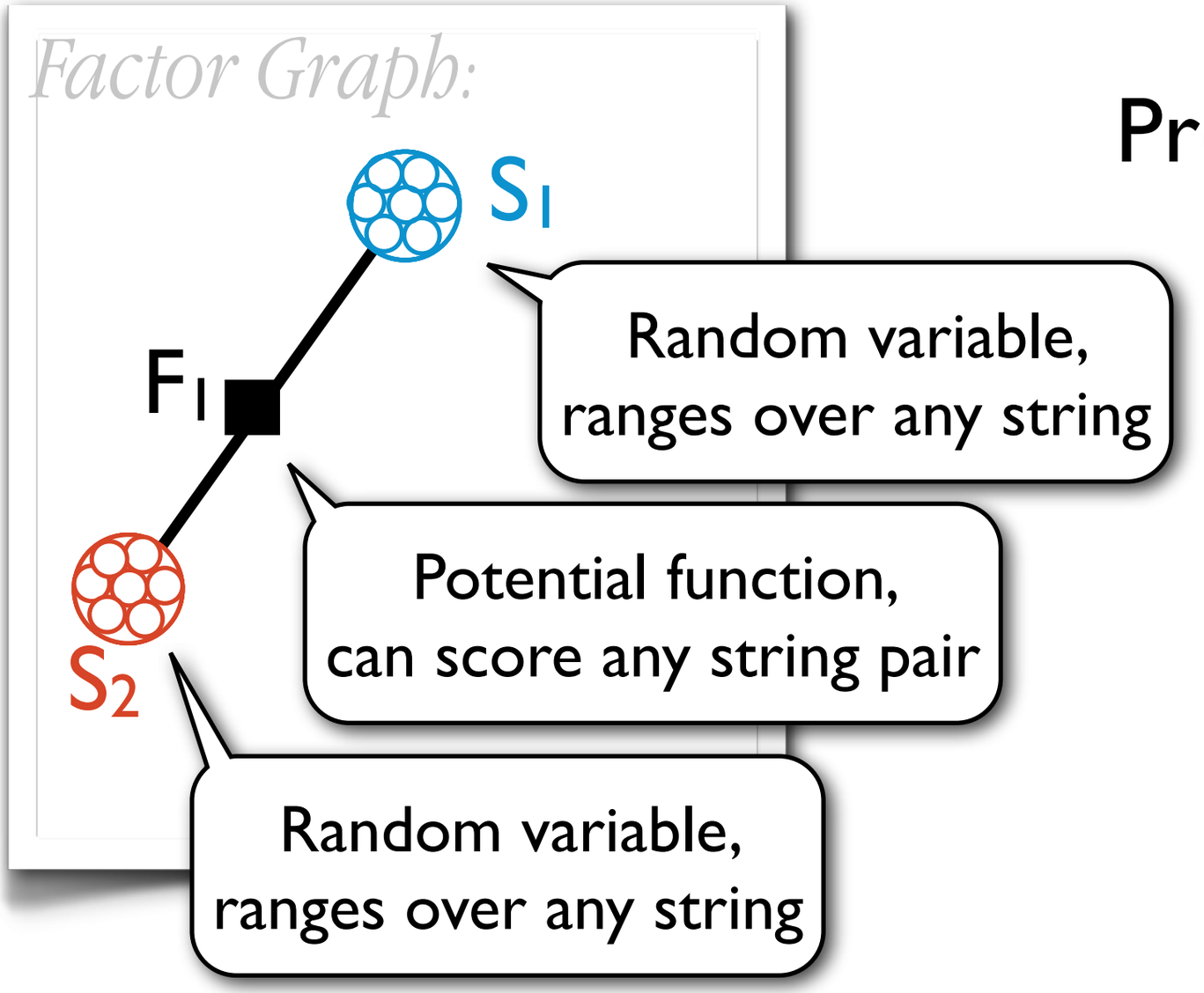
$$\Pr(s_1, s_2) = \frac{1}{Z} \times F_1(s_1, s_2)$$

2 Model. *Factor graph examples*



$$\Pr(s_1, s_2) = \frac{1}{Z} \times F_1(s_1, s_2)$$

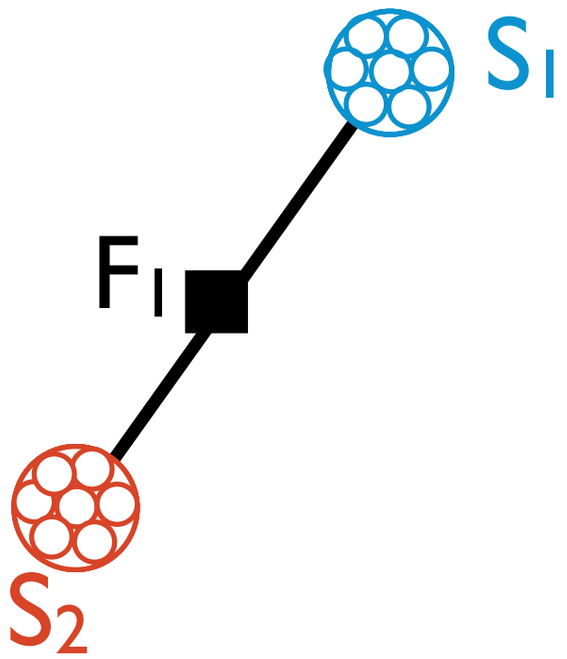
2 Model. *Factor graph examples*



$$\Pr(s_1, s_2) = \frac{1}{Z} \times F_1(s_1, s_2)$$

2 Model. *Factor graph examples*

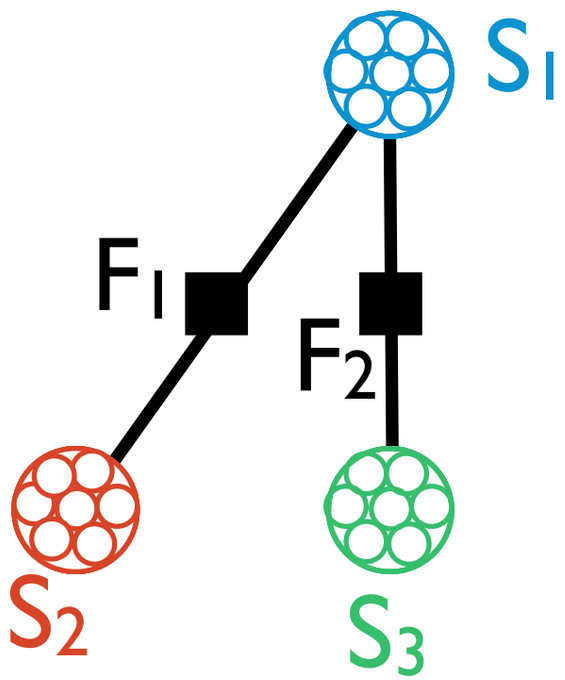
Factor Graph:



$$\Pr(s_1, s_2) = \frac{1}{Z} \times F_1(s_1, s_2)$$

2 Model. *Factor graph examples*

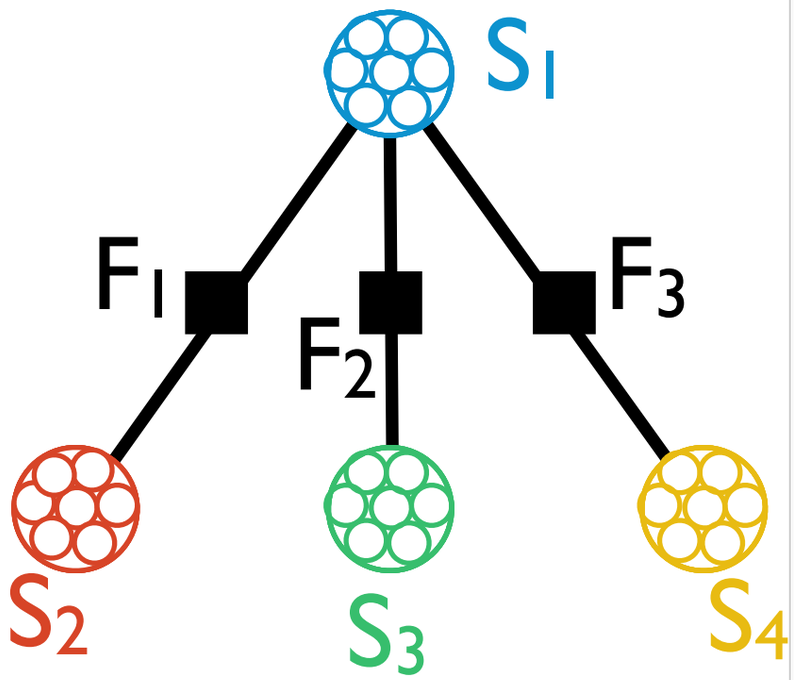
Factor Graph:



$$\Pr(s_1, s_2, s_3) = \frac{1}{Z} \times F_1(s_1, s_2) \times F_2(s_1, s_3)$$

2 Model. *Factor graph examples*

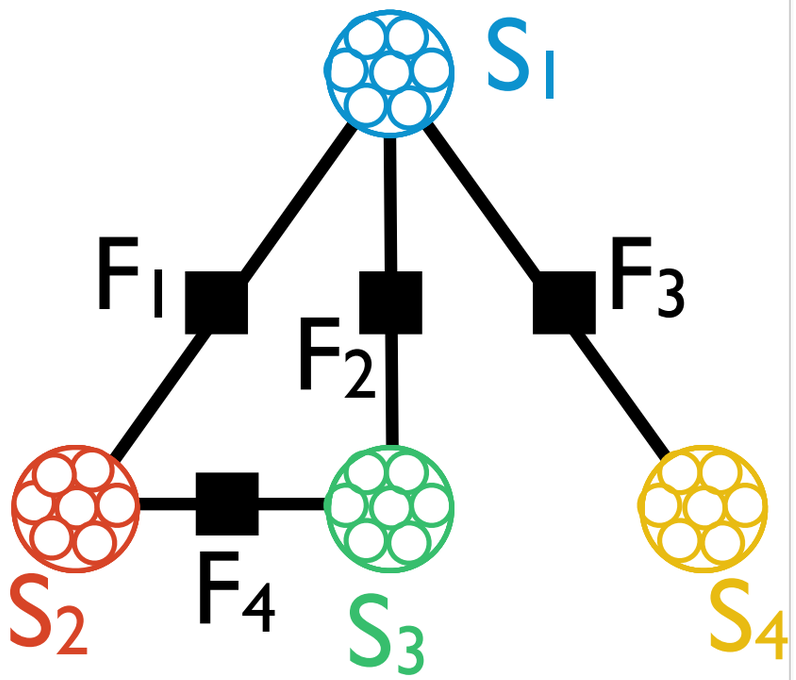
Factor Graph:



$$\Pr(s_1, s_2, s_3, s_4) = \frac{1}{Z} \times F_1(s_1, s_2) \times F_2(s_1, s_3) \times F_3(s_1, s_4)$$

2 Model. *Factor graph examples*

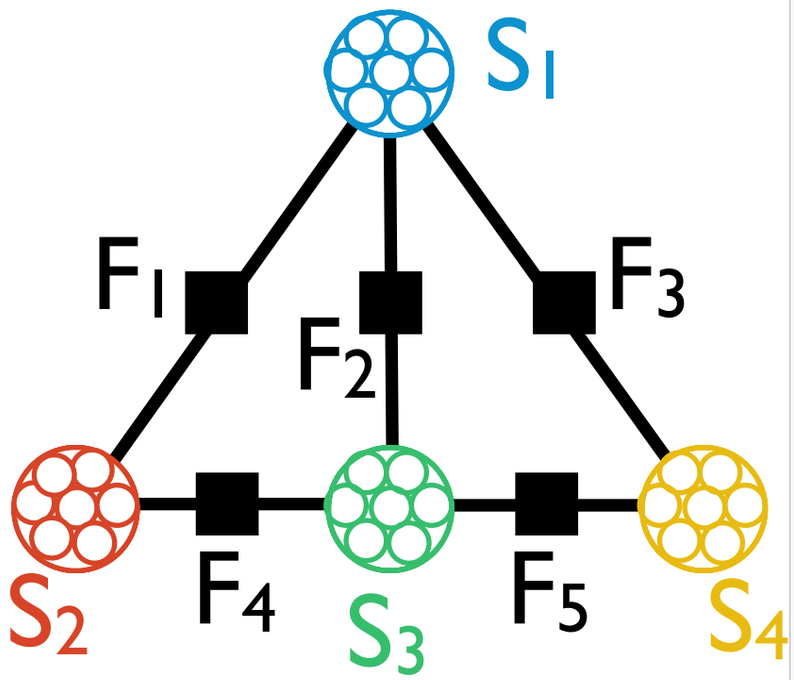
Factor Graph:



$$\Pr(s_1, s_2, s_3, s_4) = \frac{1}{Z} \times F_1(s_1, s_2) \times F_2(s_1, s_3) \times F_3(s_1, s_4) \times F_4(s_2, s_3)$$

2 Model. *Factor graph examples*

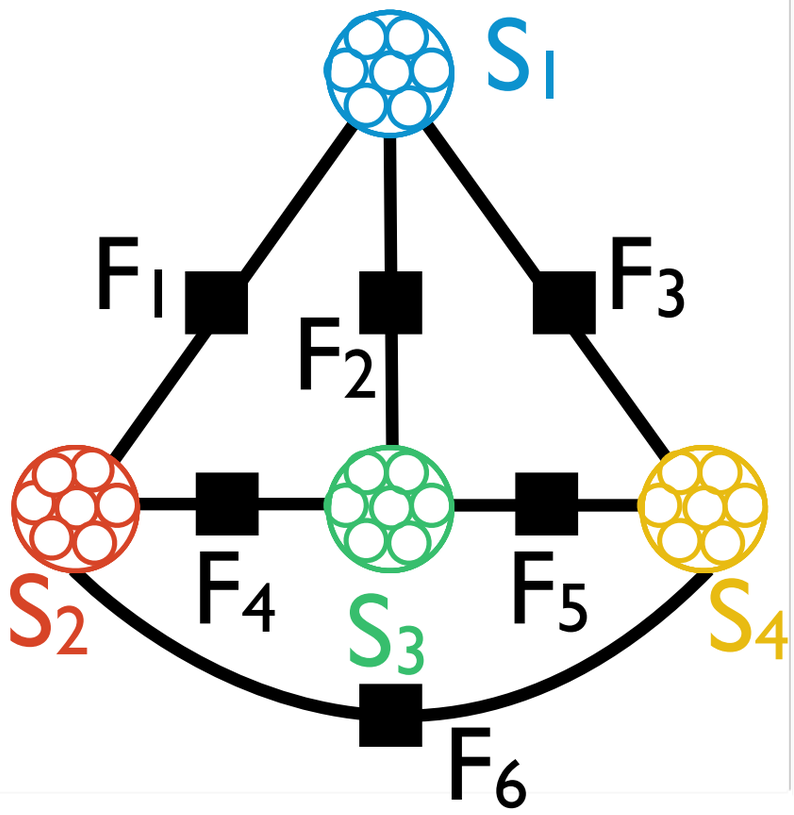
Factor Graph:



$$\Pr(s_1, s_2, s_3, s_4) = \frac{1}{Z} \times F_1(s_1, s_2) \times F_2(s_1, s_3) \times F_3(s_1, s_4) \times F_4(s_2, s_3) \times F_5(s_3, s_4)$$

2 Model. *Factor graph examples*

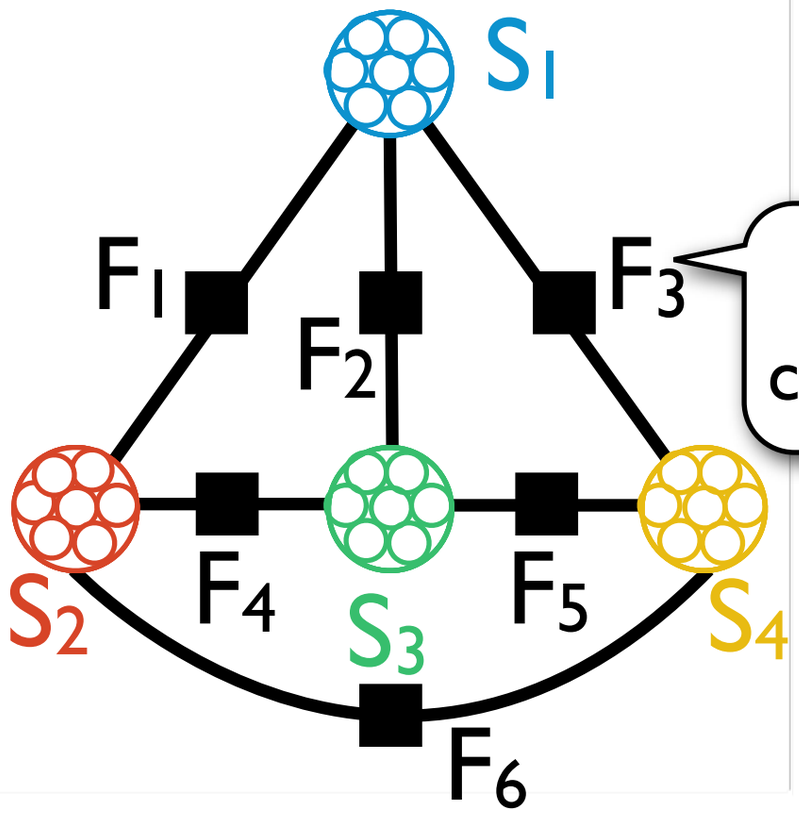
Factor Graph:



$$\Pr(s_1, s_2, s_3, s_4) = \frac{1}{Z} \times F_1(s_1, s_2) \times F_2(s_1, s_3) \times F_3(s_1, s_4) \times F_4(s_2, s_3) \times F_5(s_3, s_4) \times F_6(s_2, s_4)$$

2 Model. *Factor graph examples*

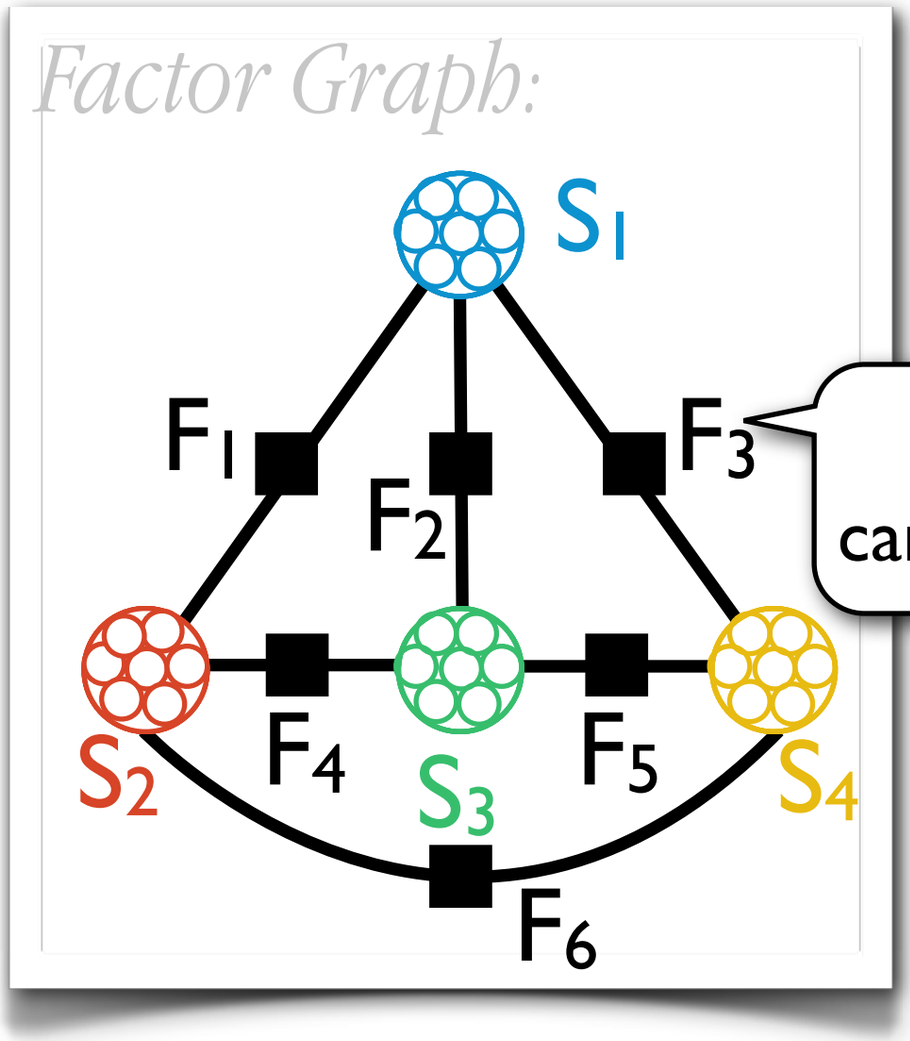
Factor Graph:



Potential function, can score any string pair

$$\Pr(s_1, s_2, s_3, s_4) = \frac{1}{Z} \times F_1(s_1, s_2) \times F_2(s_1, s_3) \times F_3(s_1, s_4) \times F_4(s_2, s_3) \times F_5(s_3, s_4) \times F_6(s_2, s_4)$$

2 Model. *Factor graph examples*

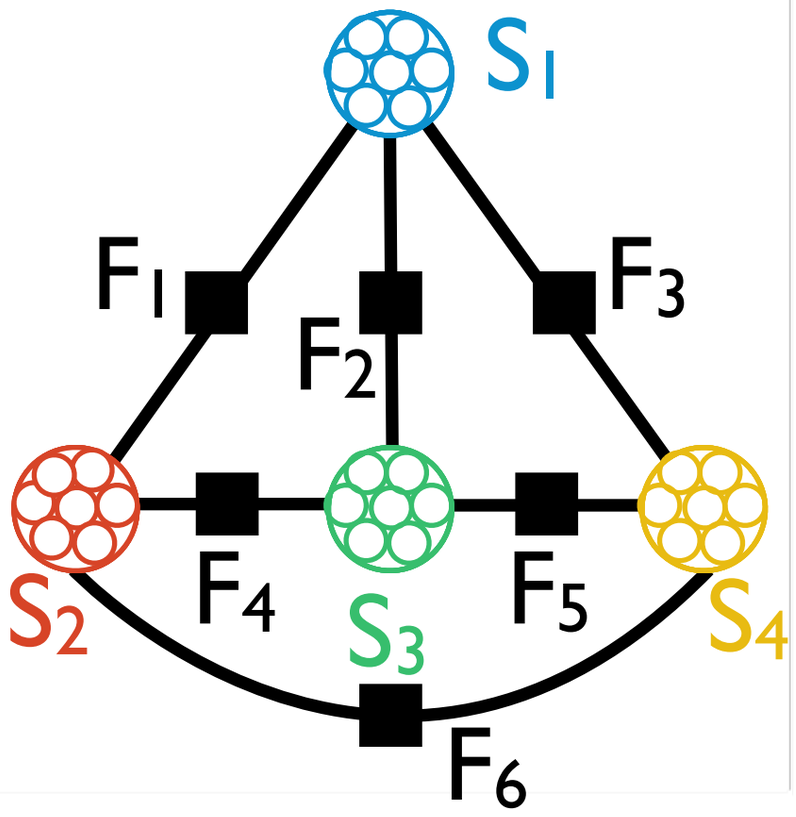


Potential function, can score any string pair

Each potential function **F** is computed by a **finite-state transducer**.

2 Model. *Factor graph examples*

Factor Graph:



$$\Pr(s_1, s_2, s_3, s_4) = \frac{1}{Z} \times F_1(s_1, s_2) \times F_2(s_1, s_3) \times F_3(s_1, s_4) \times F_4(s_2, s_3) \times F_5(s_3, s_4) \times F_6(s_2, s_4)$$

A formal description of such a model ...

2 Model. *Summary*

- It is formally an undirected **graphical model** (a.k.a. Markov Random Field, *MRF*),
- in which the **variables are string-valued**, and the **factors** (potential functions) **are finite-state transducers.**

Dreyer & Eisner, 2009

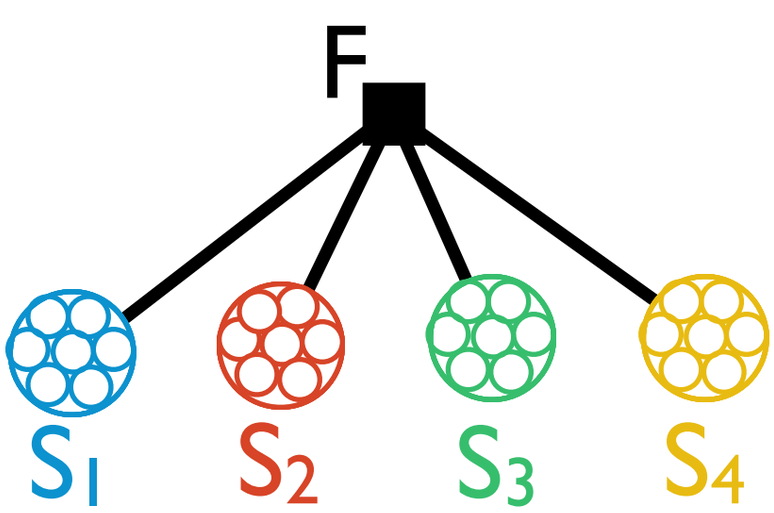
2 Model. *Less formal description*

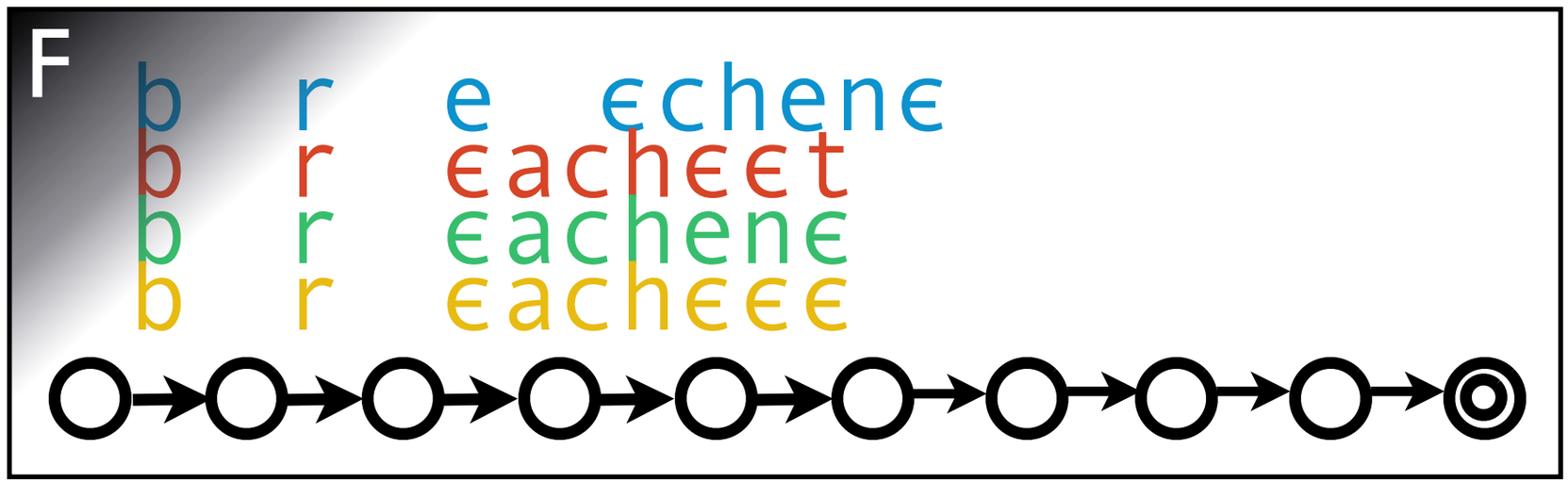
To **model multiple strings** and their various interactions, I ...

- use many **finite-state transducers**,
- have each of them look at a different string pair,
- **plug them together** into a big network,
- and **coordinate them** to predict all strings jointly (also: *train* the transducers jointly).

2 Model. *Comparison with k-tape FSM*

- Model k strings with a k-tape finite-state machine?





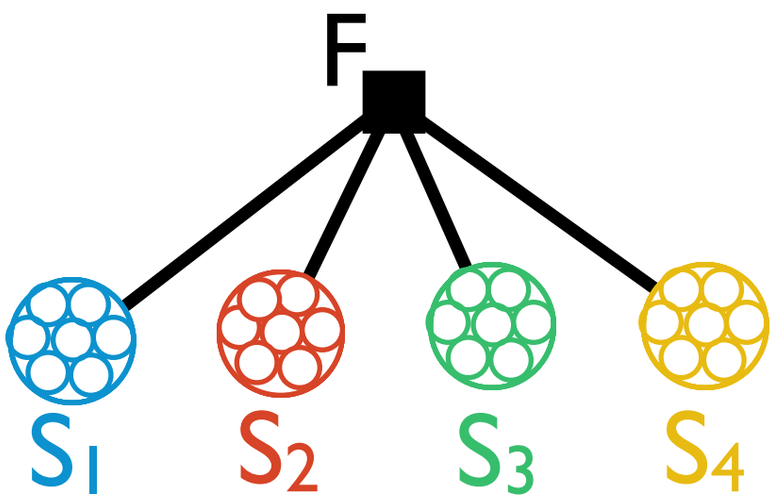
- Factored model more powerful:

☺ Encode swaps and other useful models

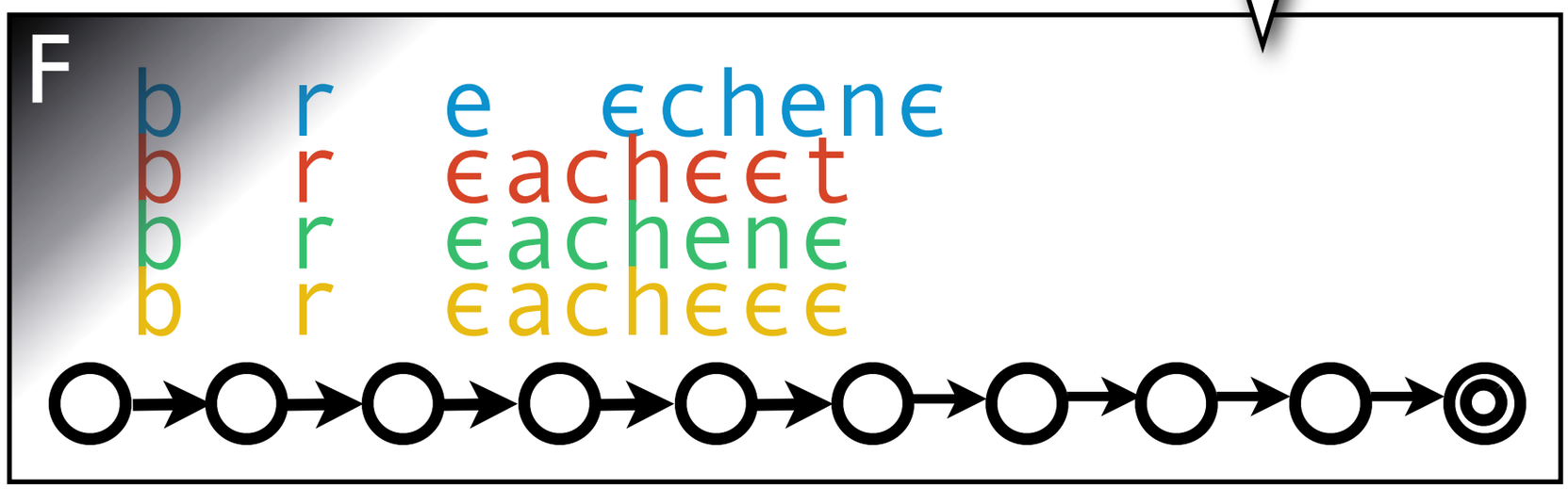
☹ Encode undecidable models

② Model. *Comparison with k-tape FSM*

- Model k strings with a k-tape finite-state machine?
- $>26^k$ arcs, intractable!



Multiple-sequence alignment



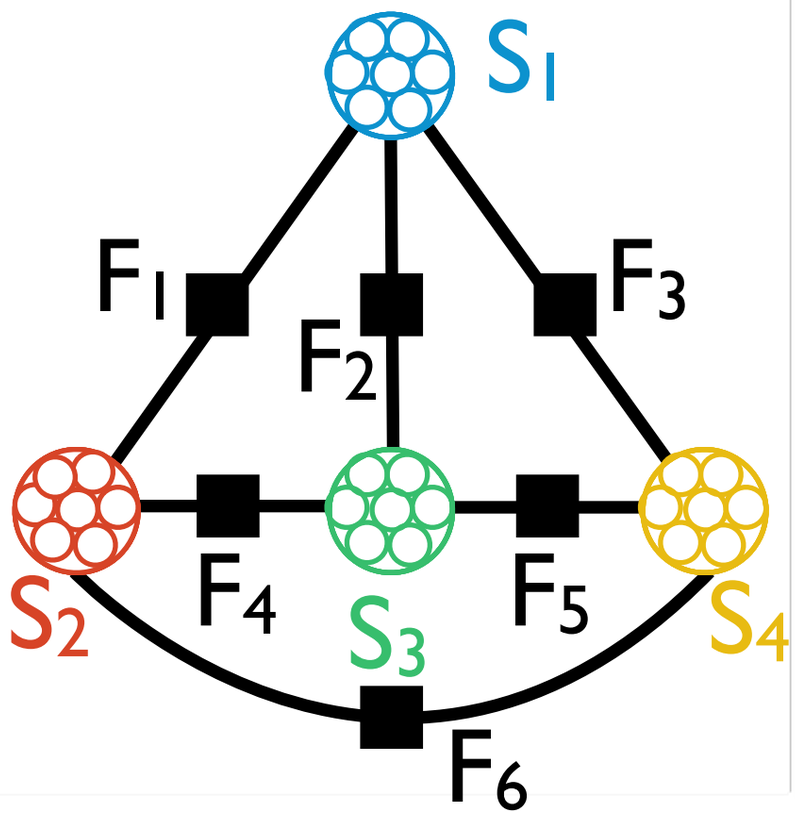
- Factored model more powerful:

☺ Encode swaps and other useful models

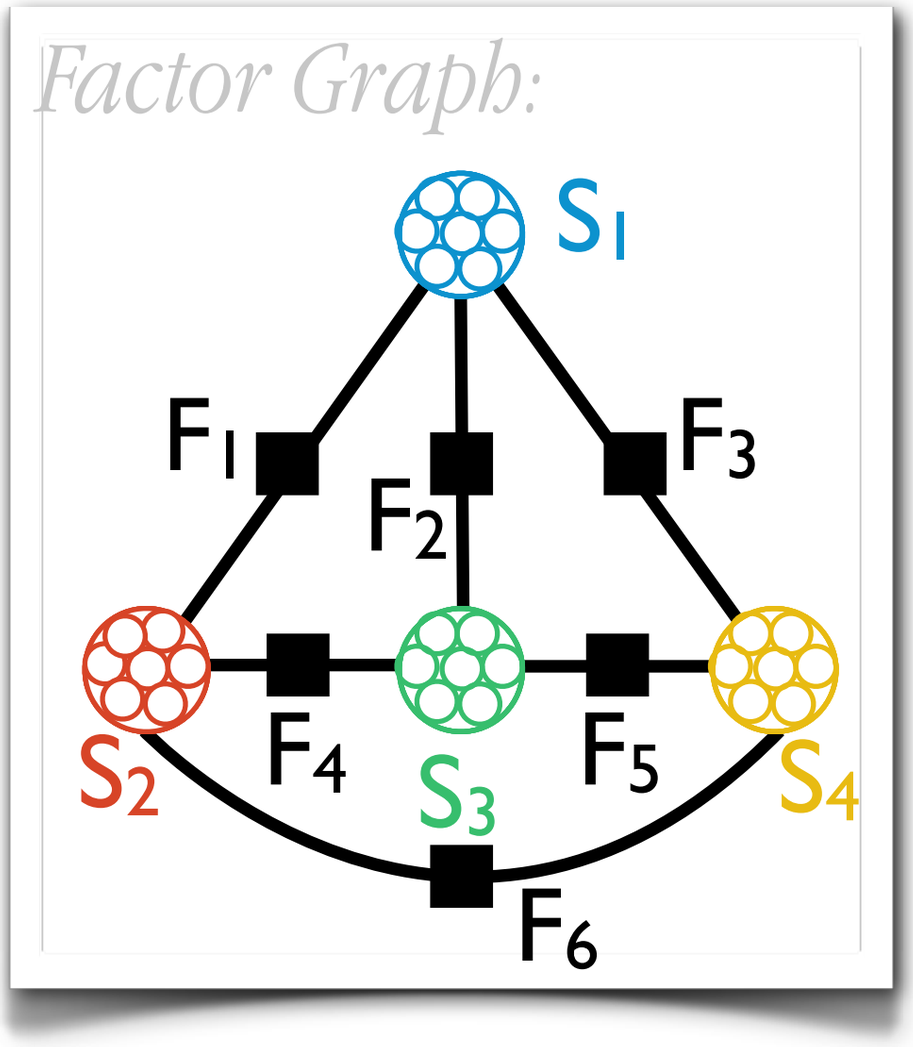
☹ Encode undecidable models

2 Inference. *Overview*

Factor Graph:

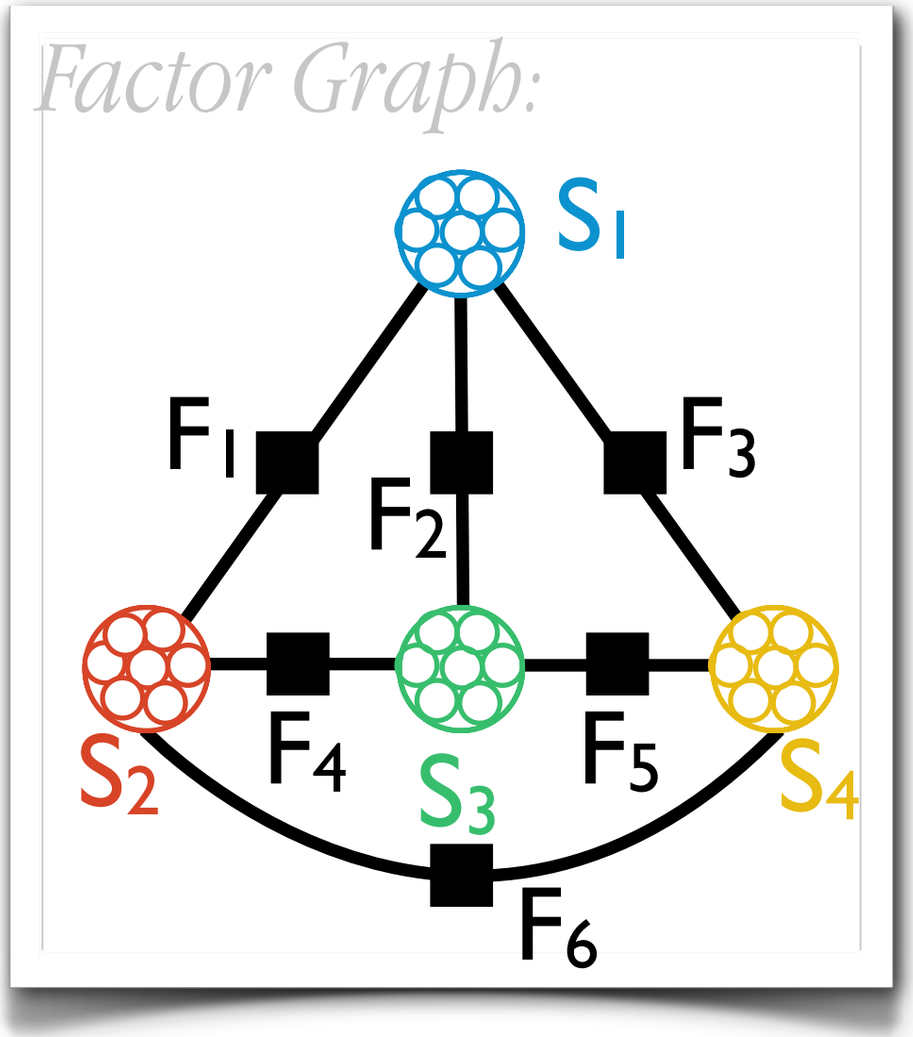


2 Inference. *Overview*



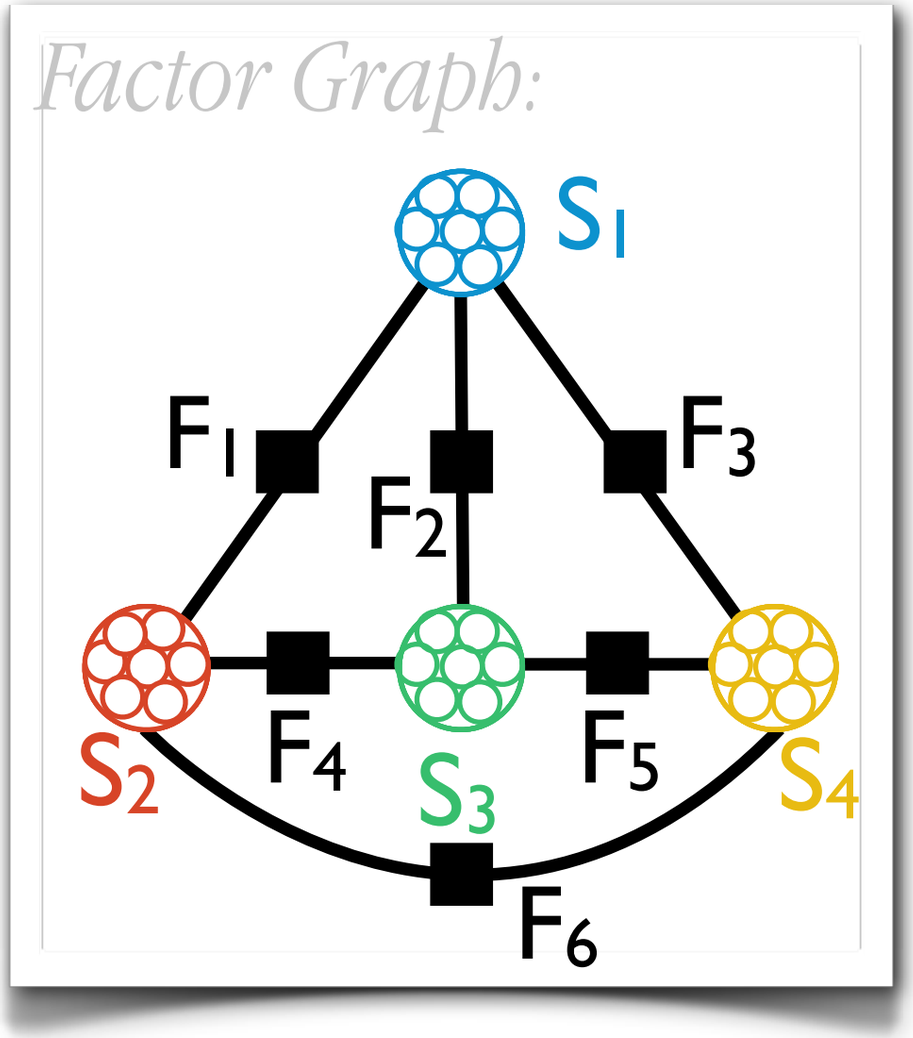
- Run **Belief Propagation (BP)**

2 Inference. *Overview*



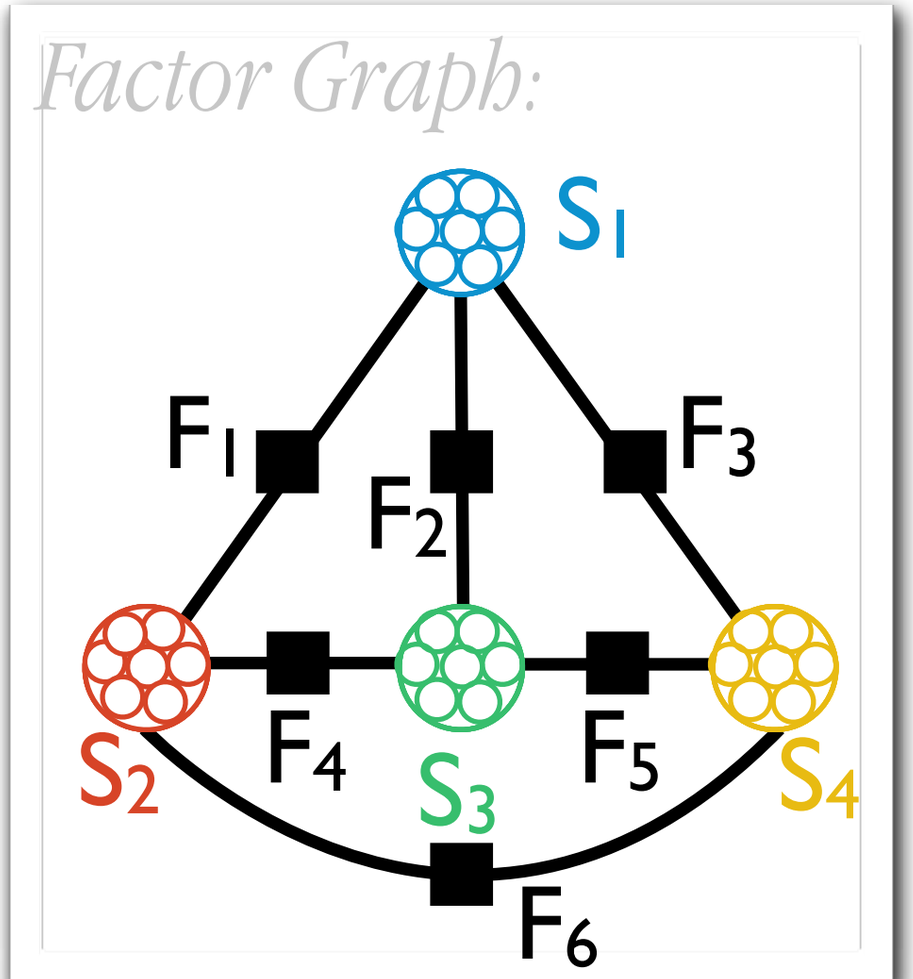
- Run **Belief Propagation (BP)**
- BP is a message-passing algorithm, a **generalization of forward-backward.**

2 Inference. *Overview*



- Run **Belief Propagation (BP)**
- BP is a message-passing algorithm, a **generalization of forward-backward**.
- BP computes marginals

2 Inference. *Overview*

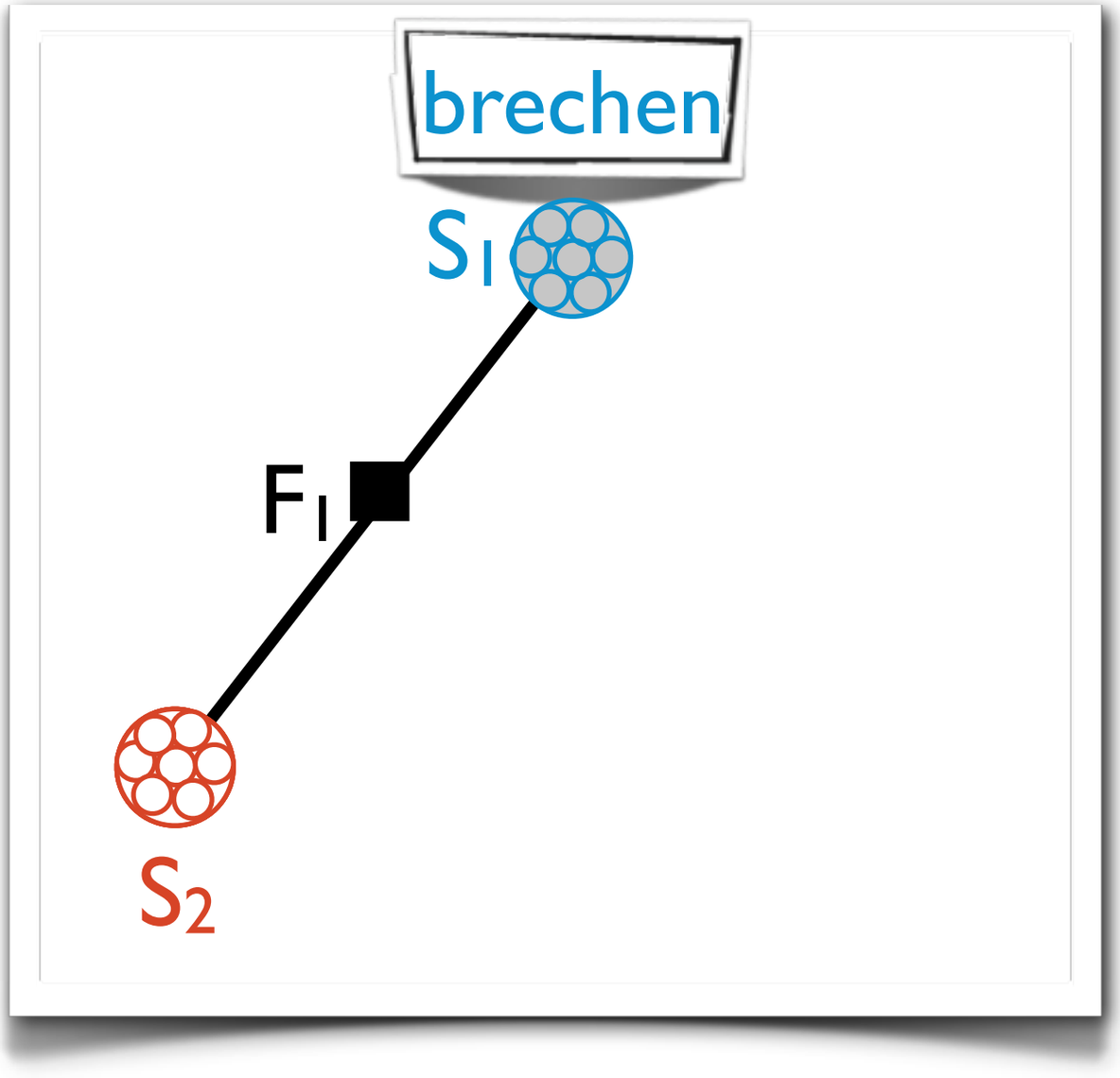


- Run **Belief Propagation (BP)**
- BP is a message-passing algorithm, a **generalization of forward-backward**.
- BP computes marginals

In our version of BP, **all messages and beliefs are finite-state machines**, which is novel.

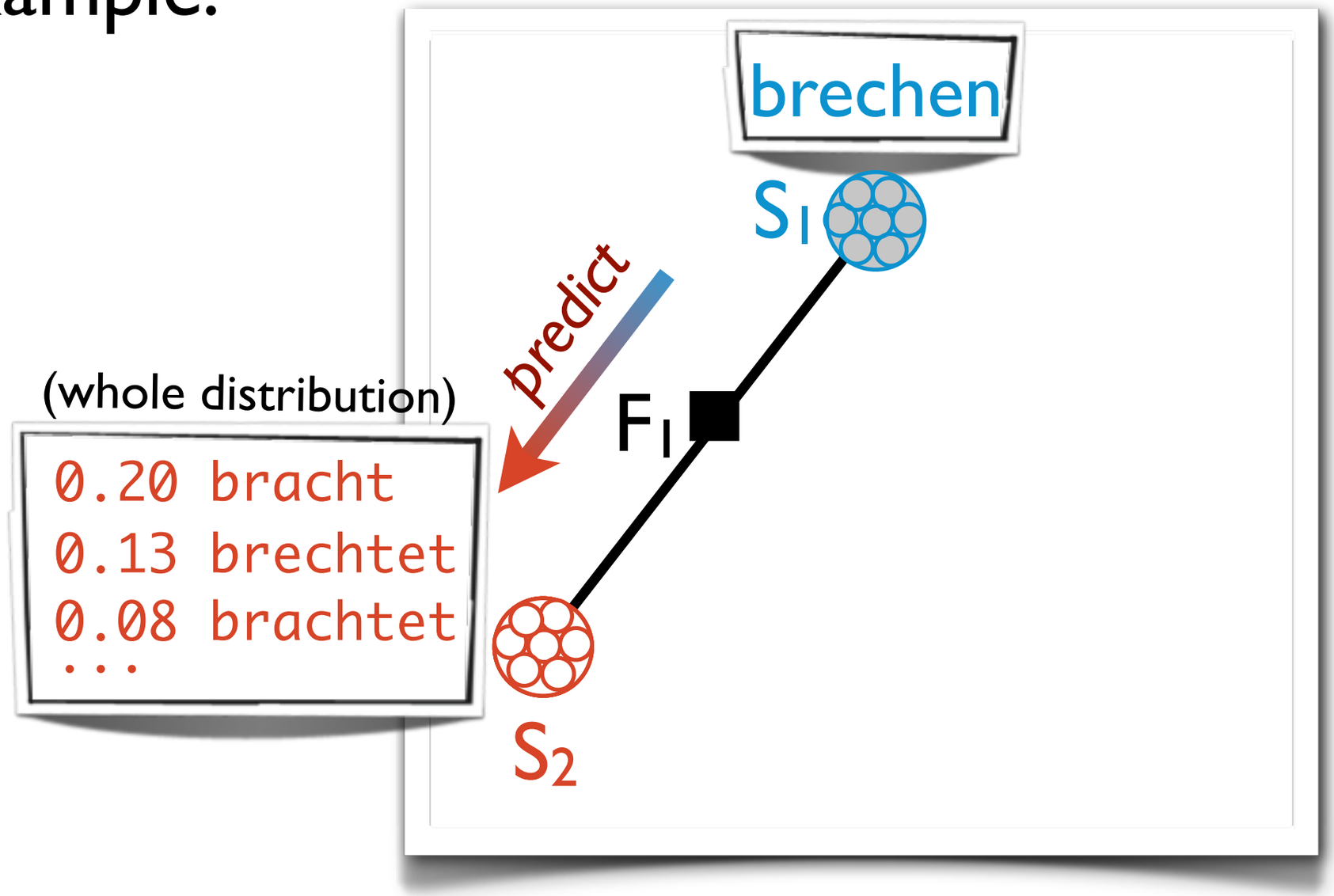
2 Inference. *Multiple strings*

Example:



2 Inference. *Multiple strings*

Example:

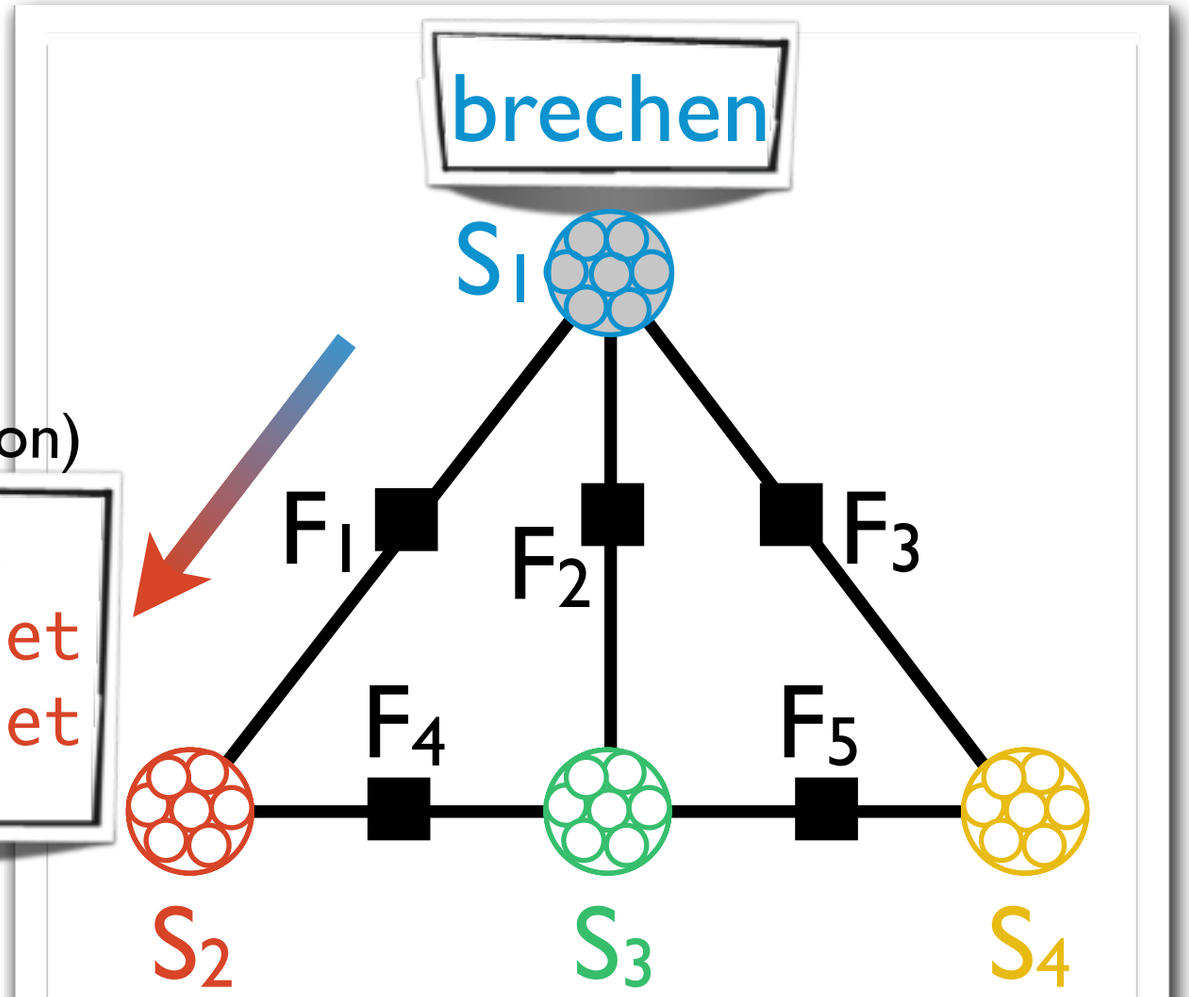


2 Inference. *Multiple strings*

Example:

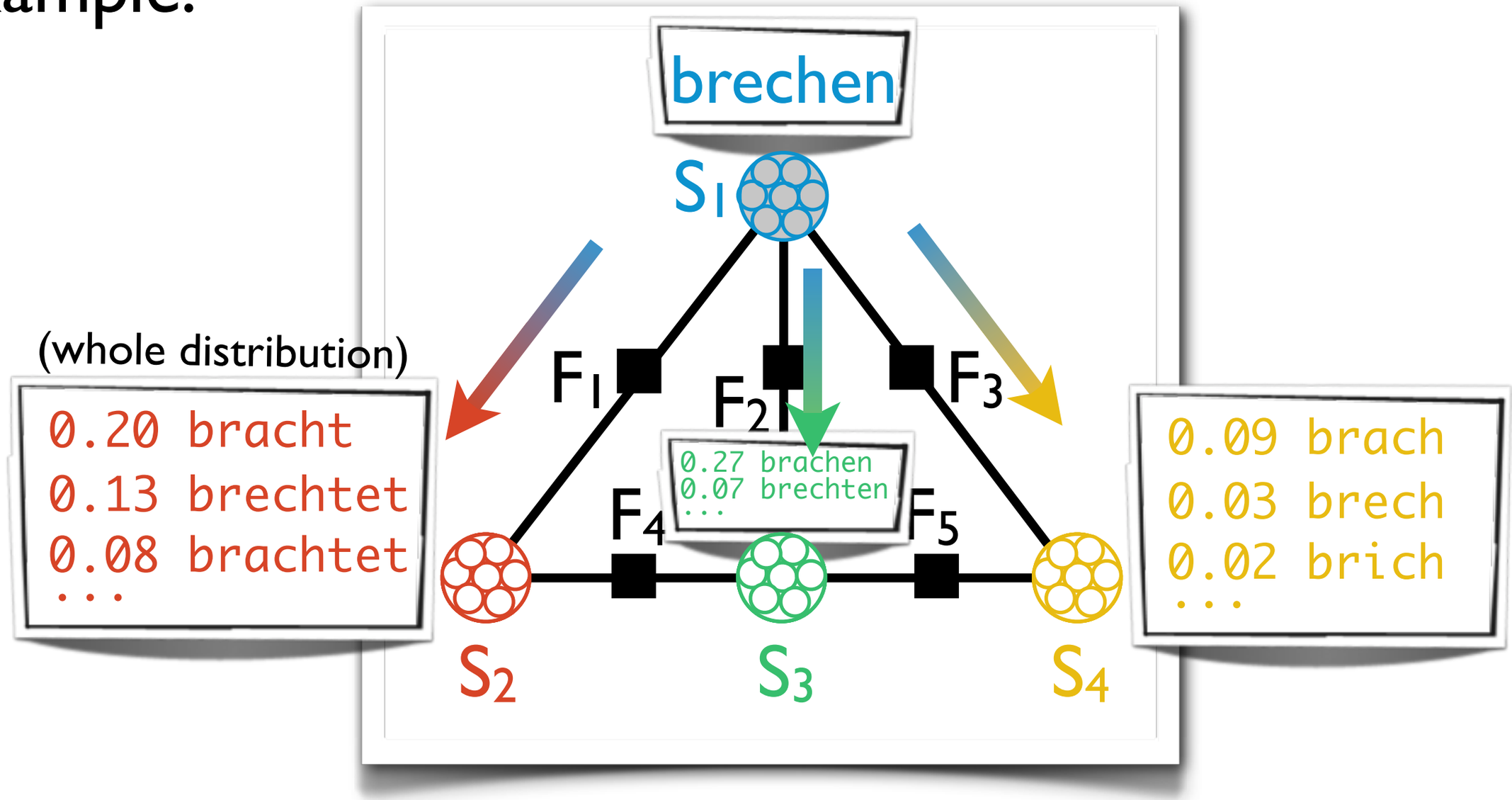
(whole distribution)

0.20 bracht
0.13 brechtet
0.08 brachtet
...



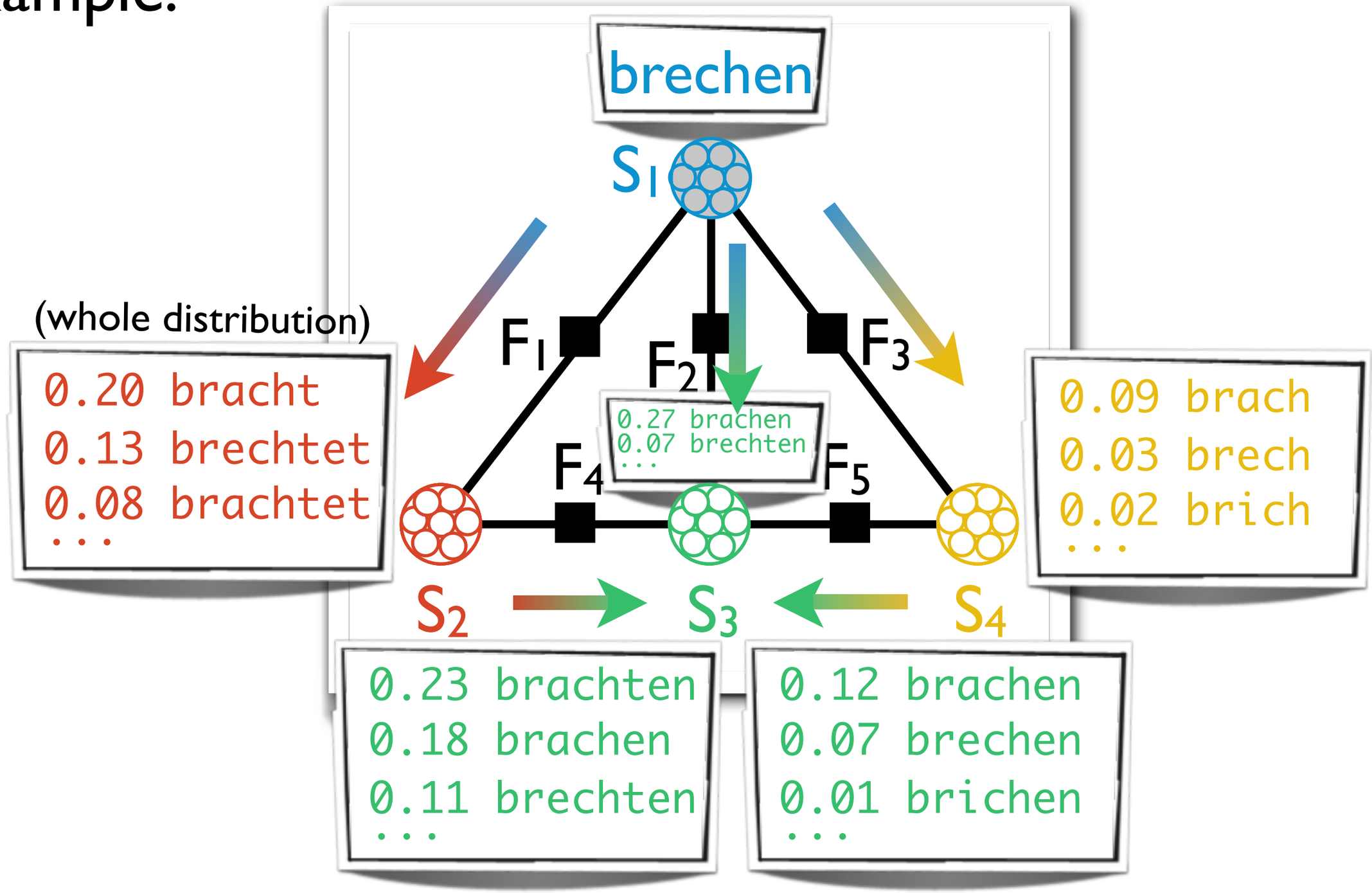
2 Inference. *Multiple strings*

Example:



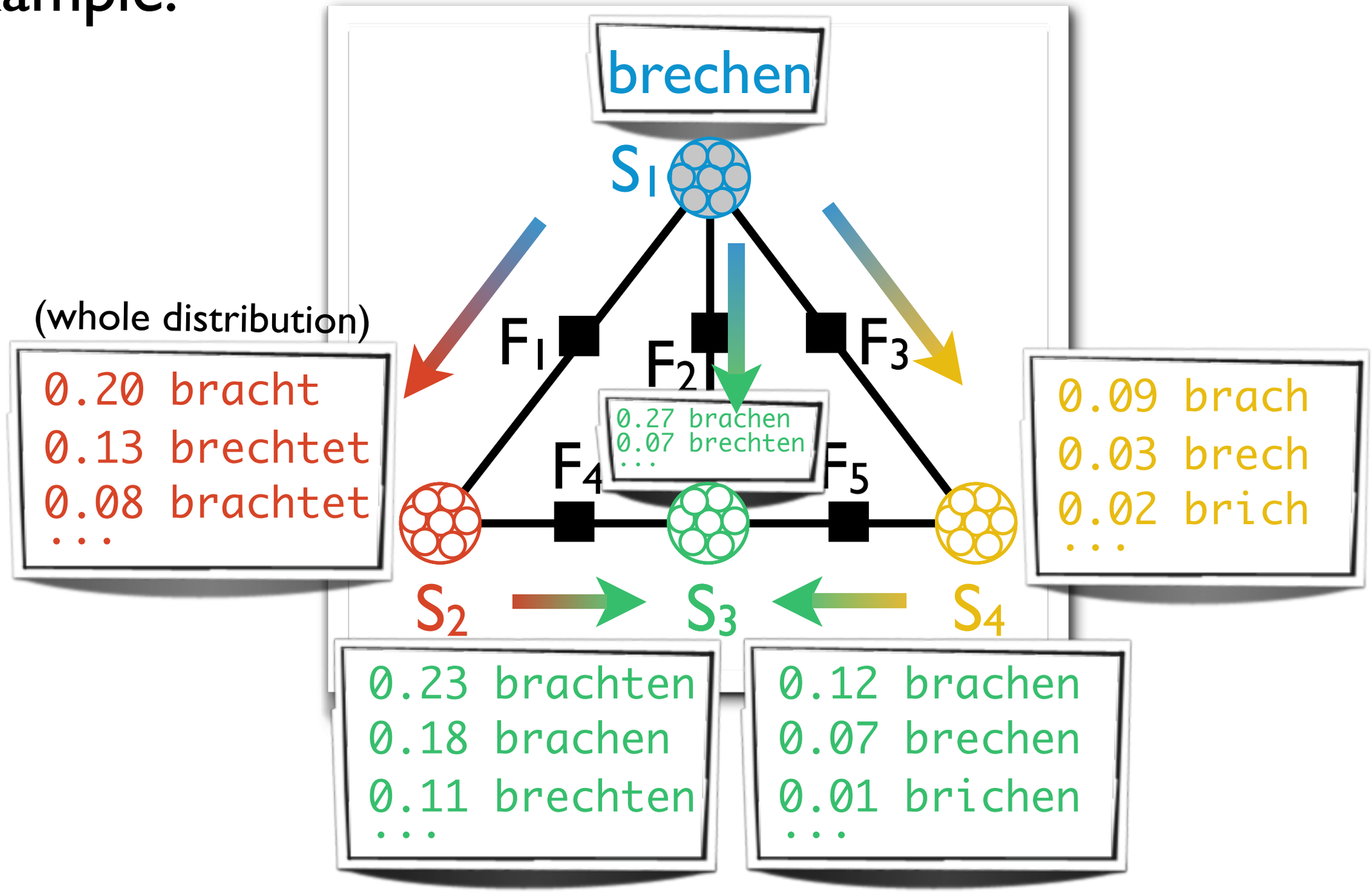
2 Inference. *Multiple strings*

Example:



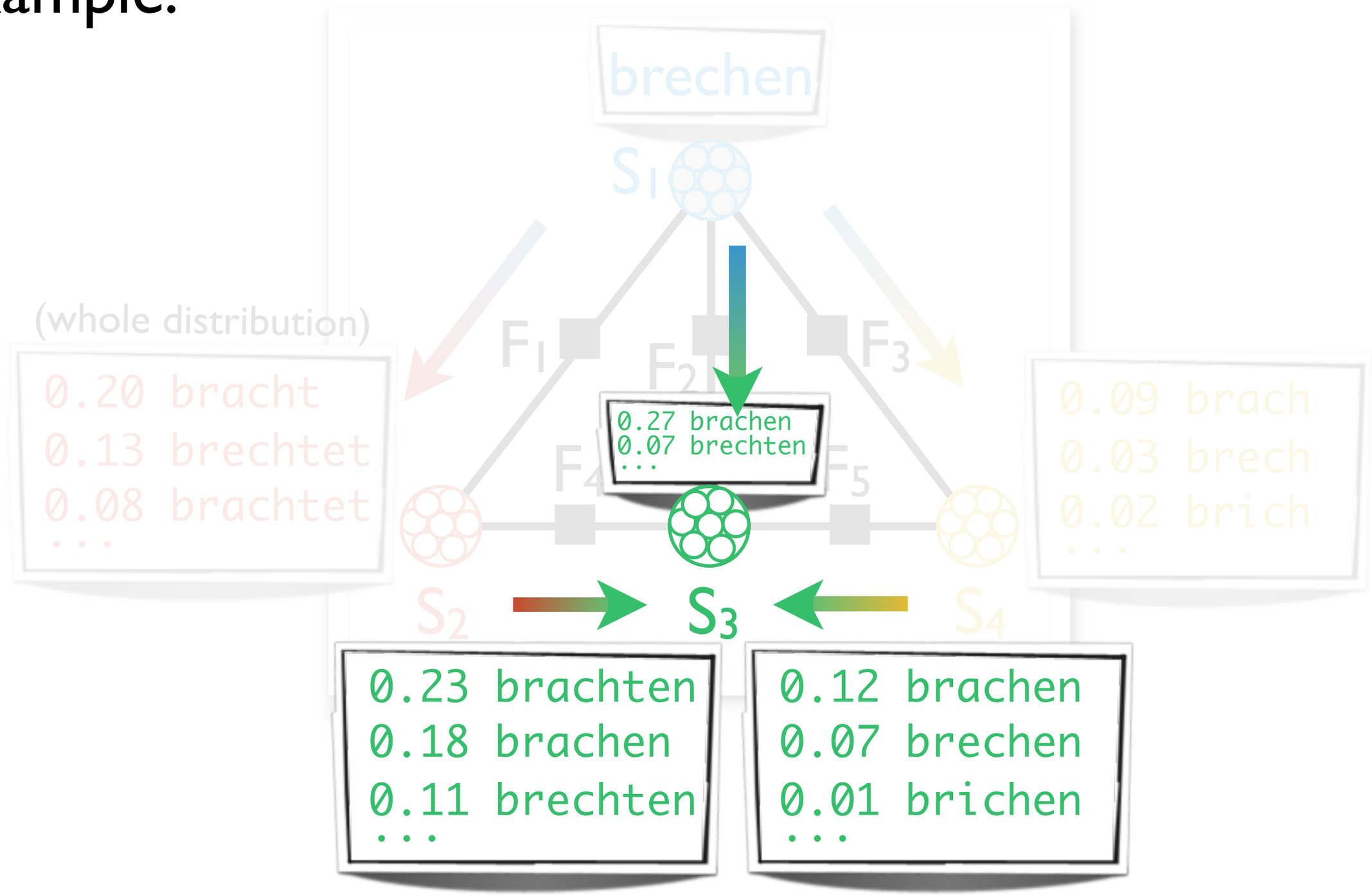
2 Inference. *Multiple strings*

Example:



2 Inference. *Multiple strings*

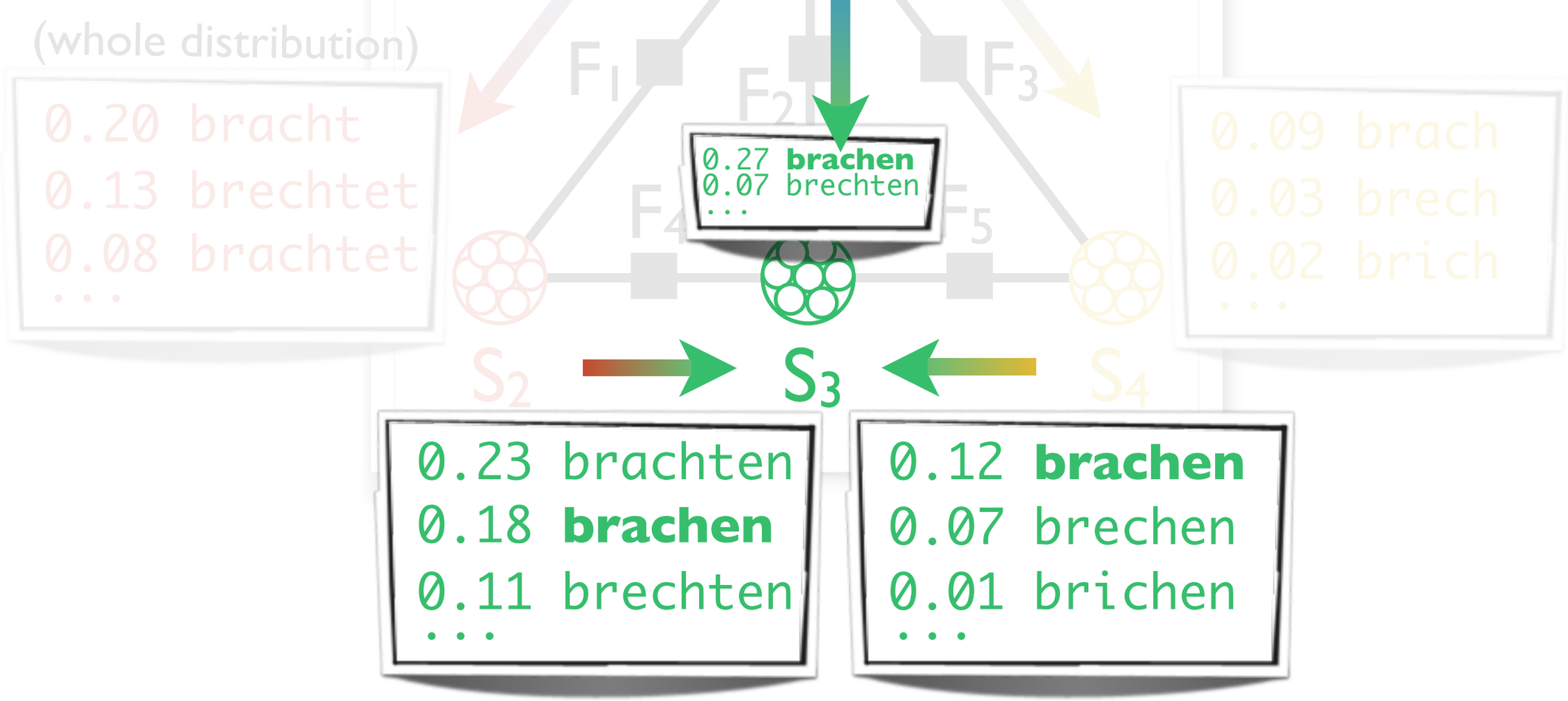
Example:



2 Inference. *Multiple strings*

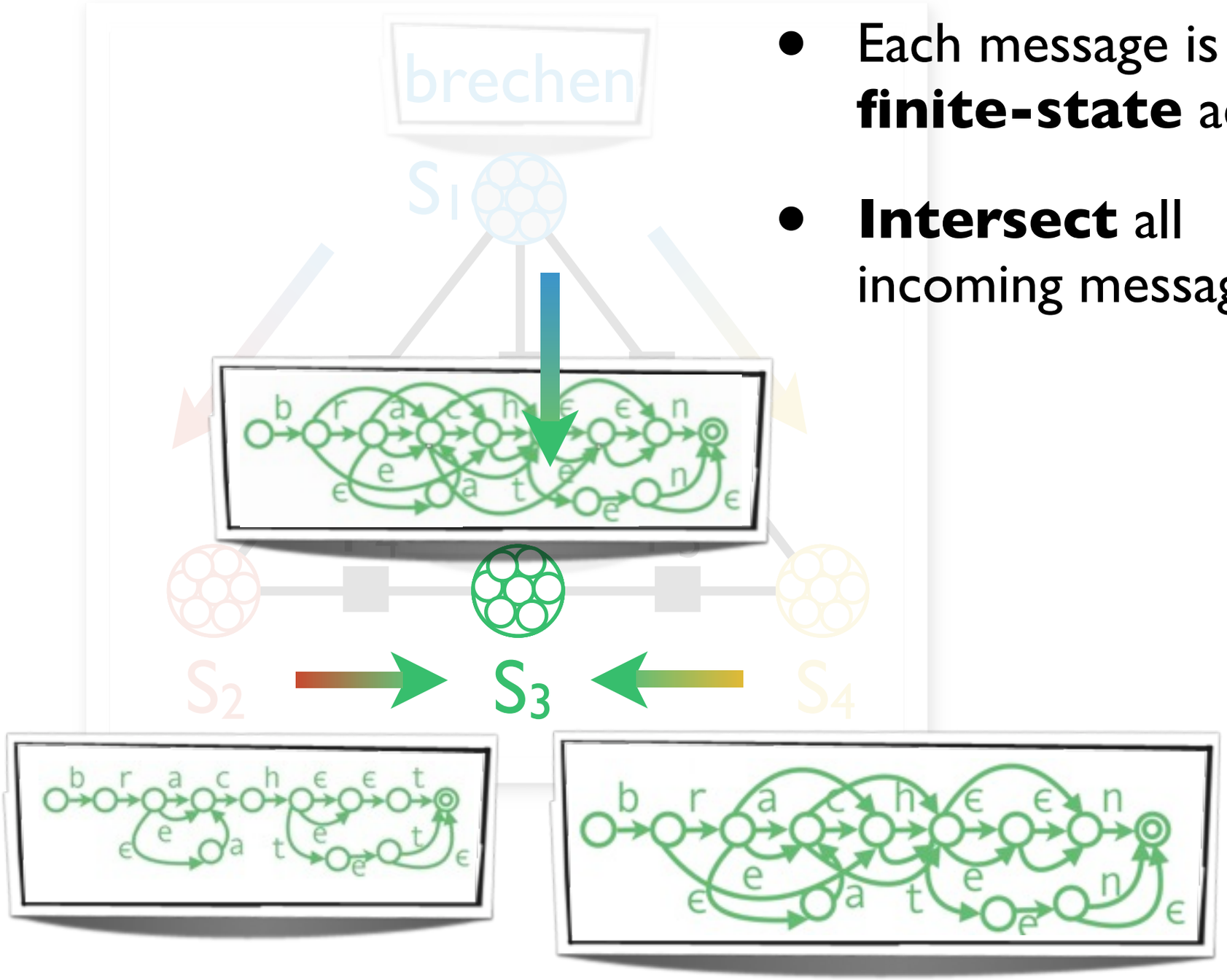
Example:

Decoding output for S_3 (consensus): **brachen**



2 Inference. *Multiple strings*

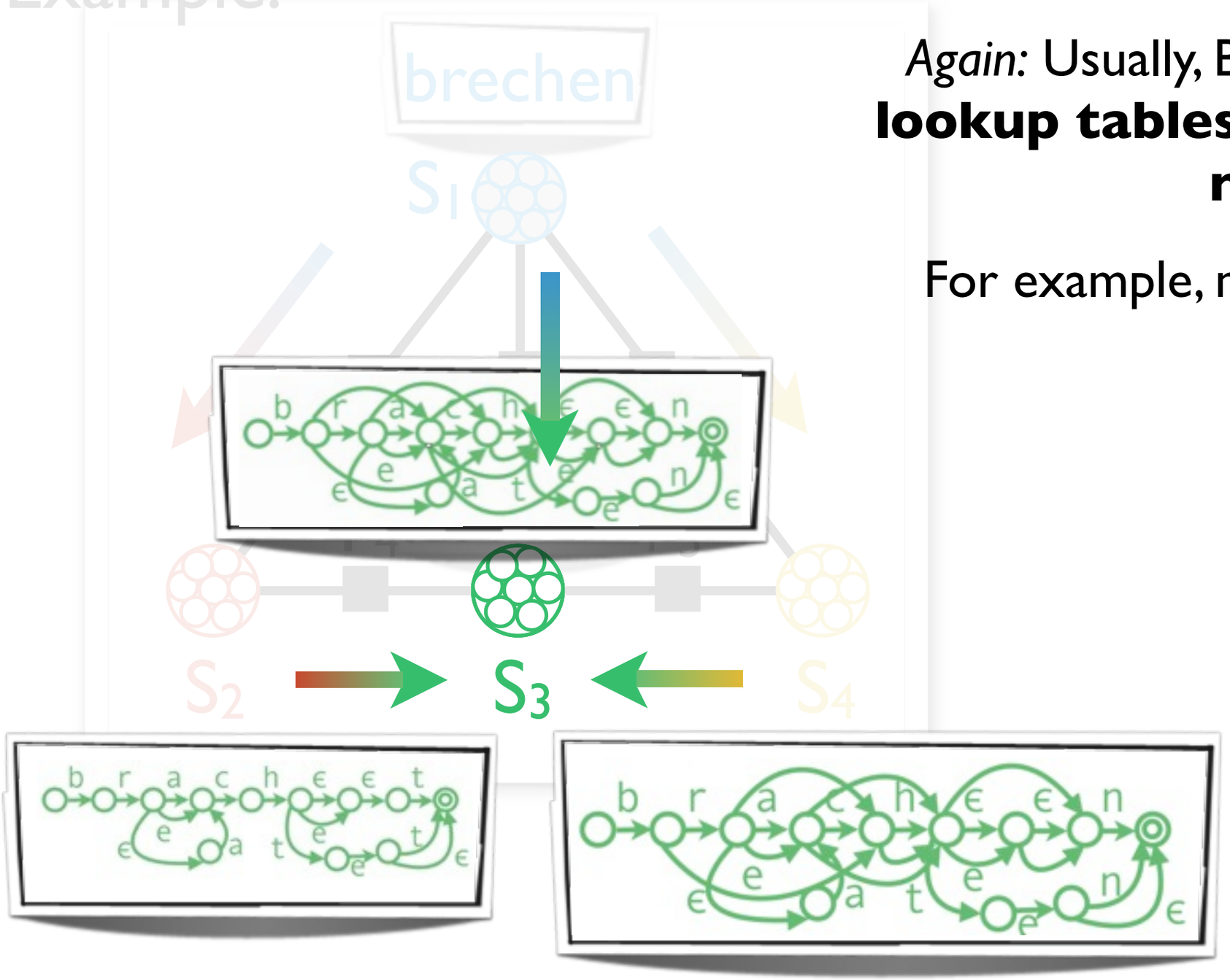
Example:



- Each message is a **finite-state** acceptor
- **Intersect** all incoming messages

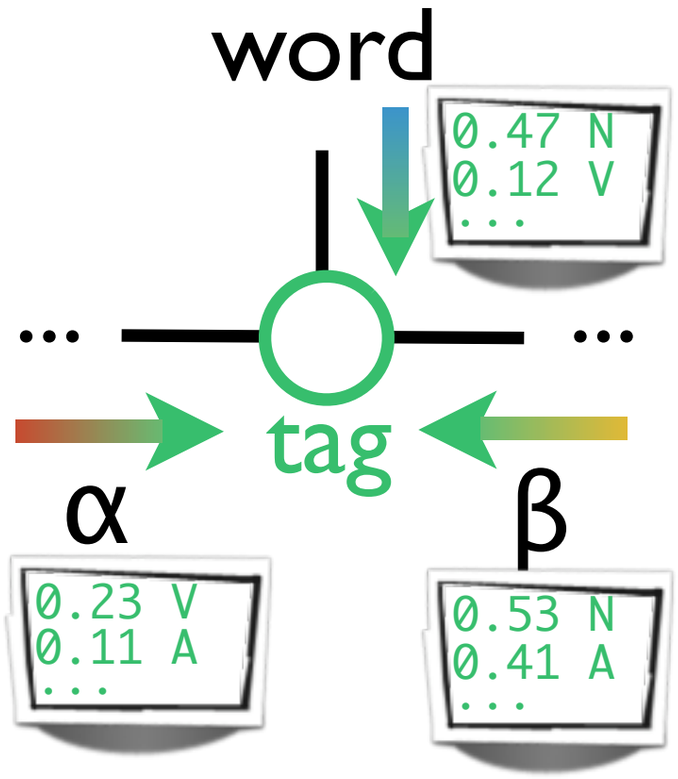
2 Inference. *Multiple strings*

Example:



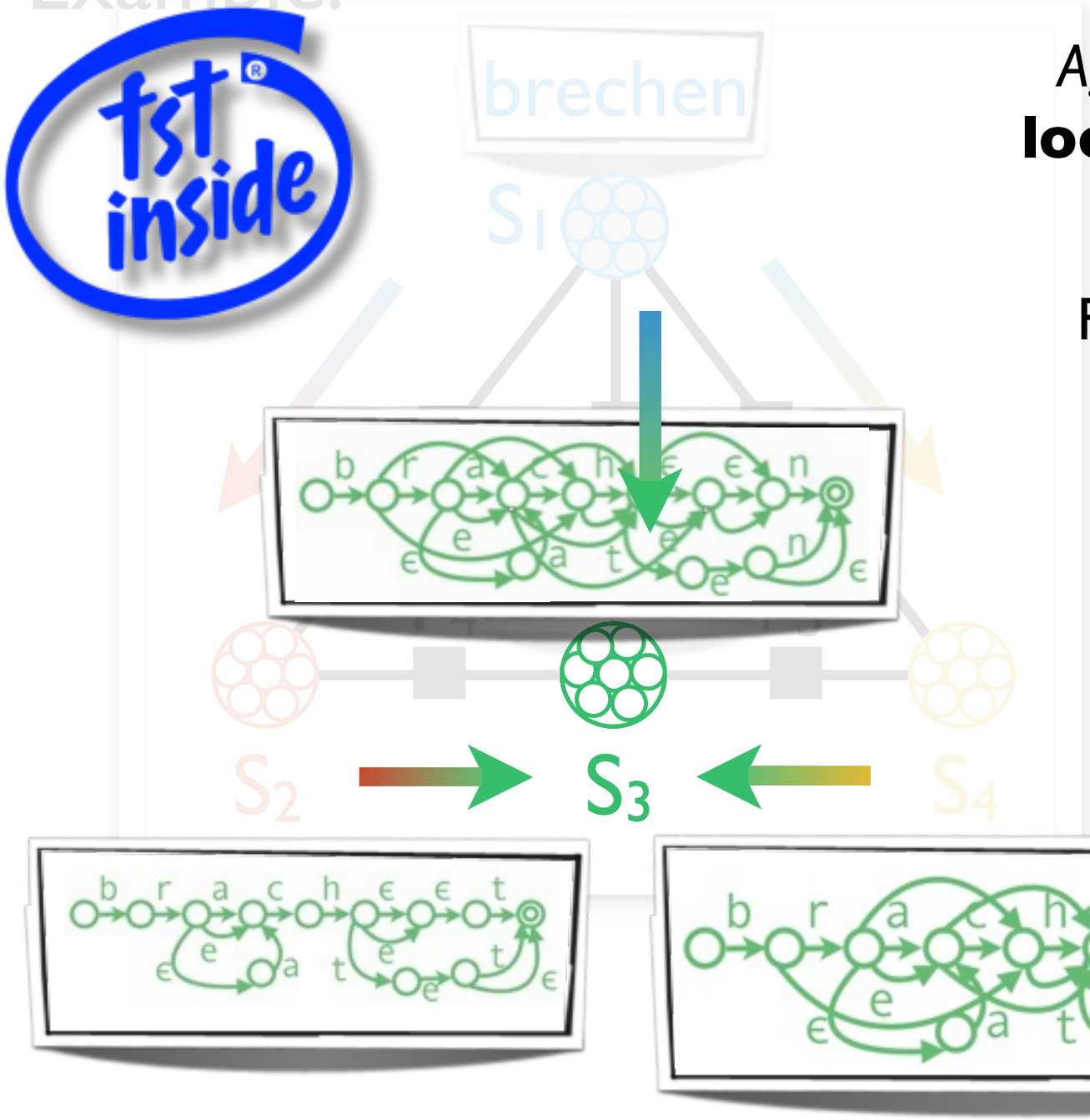
Again: Usually, BP just works with simple **lookup tables** as factors and messages, **not** finite-state machines.

For example, message passing in a CRF:



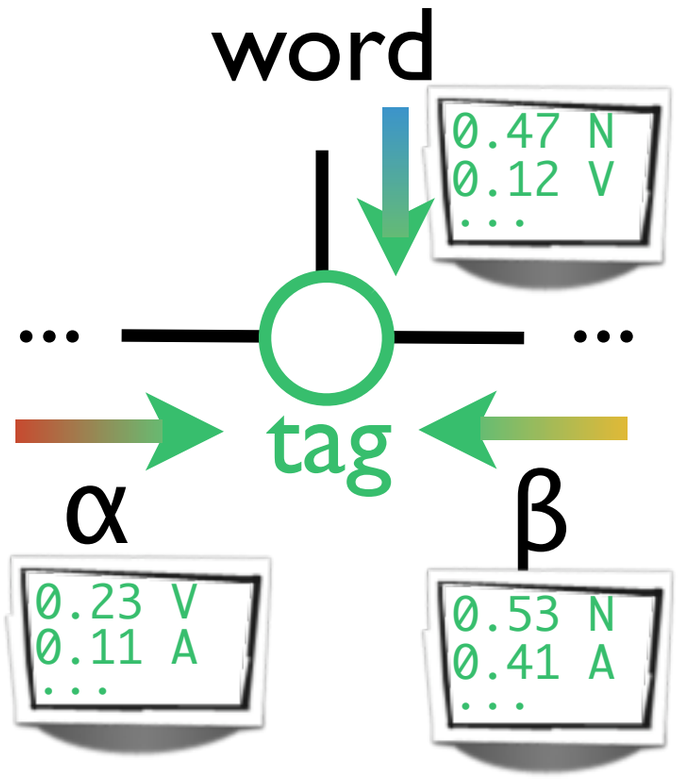
2 Inference. *Multiple strings*

Example:

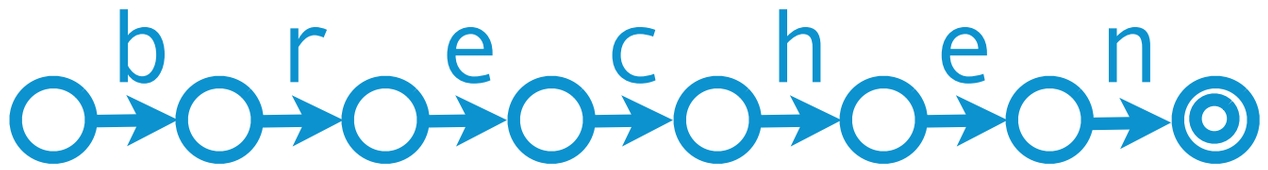


Again: Usually, BP just works with simple **lookup tables** as factors and messages, **not** finite-state machines.

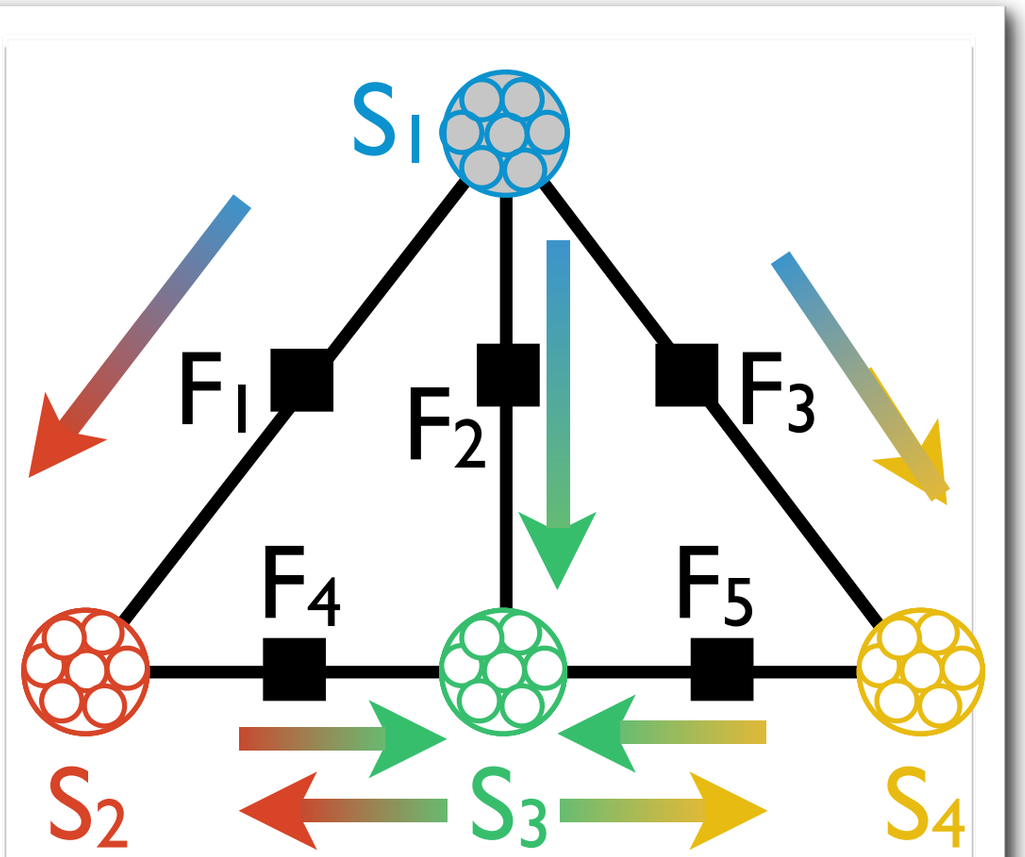
For example, message passing in a CRF:



2 Inference. *Multiple strings*

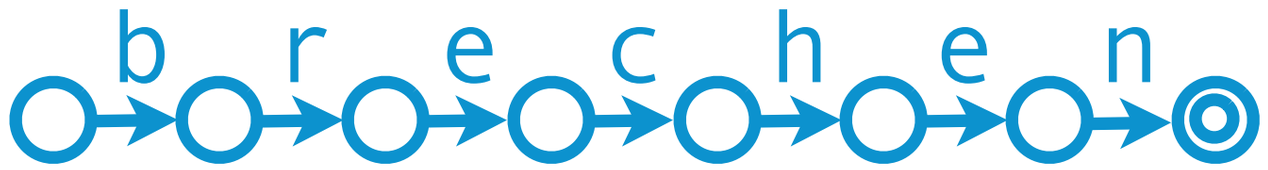
S_1 = 

(observed)

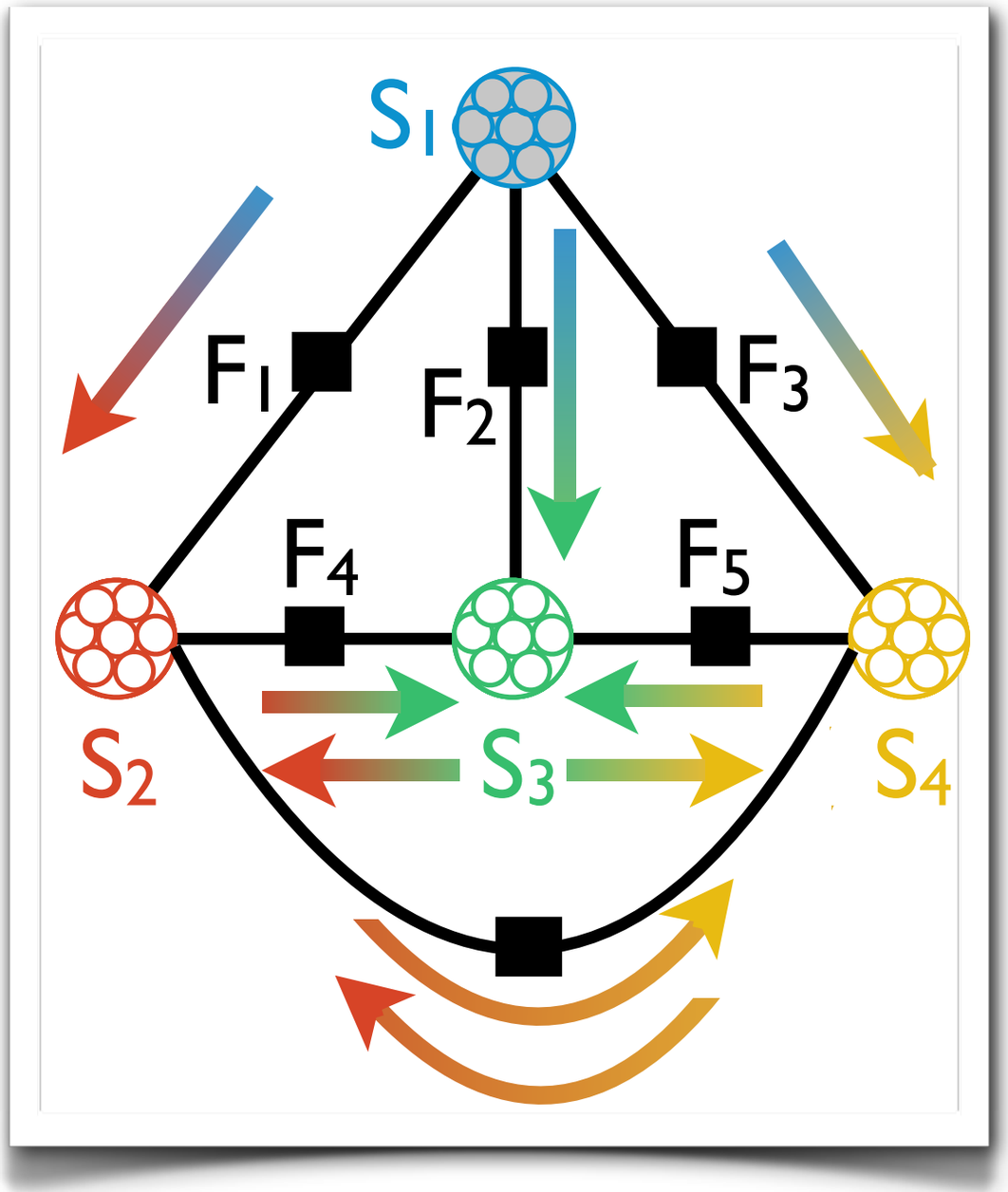


- We can also run **loopy belief propagation** on these finite-state Markov Random Fields (MRFs)
- Just **iterate** the message passing
- Issues with **intractability**, see my thesis

2 Inference. *Multiple strings*

S_1 = 

(observed)



- We can also run **loopy belief propagation** on these finite-state Markov Random Fields (MRFs)
- Just **iterate** the message passing
- Issues with **intractability**, see my thesis

2 Inference. *Multiple strings*

- Joint inference can be used to train these models from data
- Training data consists of complete or incomplete tables of forms
- We present a method to induce factor graphs in a data-driven way
- See my thesis for the approach and results

2 Conclusions / Contributions

- Presented general, novel **joint probability model** over multiple strings
- Combines **NLP techniques** *(finite-state machines)* with **machine-learning techniques** *(graphical models)*
- Markov Random Field over strings (variables: *string-valued*, **potential functions: *finite-state machines***)

2 Conclusions / Contributions

- Novel variant of belief propagation with **finite-state messages**
- Presented **approximations**
- Presented novel way of **structure induction** for string-based models based on edit distance
- Achieved significant improvements through **staged joint training** of complex factor graphs

Overview

1. String pairs

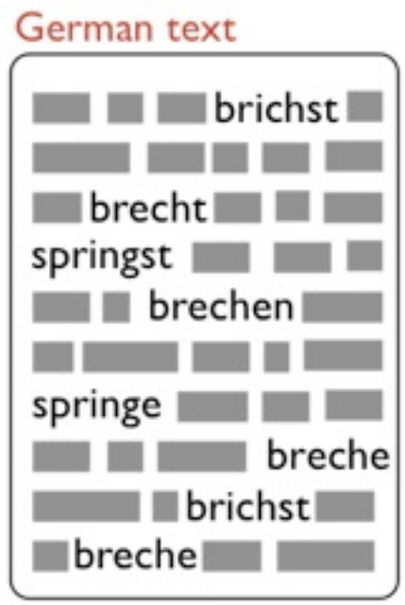
$$p\left(\begin{array}{|c|c|}\hline \text{infinitive} & \text{brechen} \\ \hline \text{1st} & \text{breche?} \\ \hline \end{array}\right)$$

2. Multiple strings (paradigms)

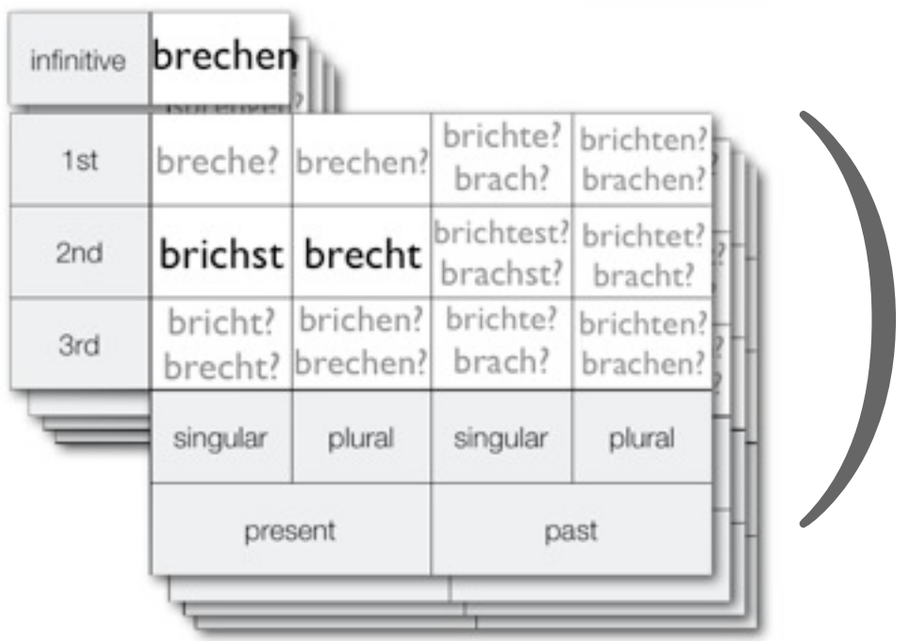
$$p\left(\begin{array}{|c|c|c|c|c|}\hline \text{infinitive} & \text{brechen} & & & \\ \hline \text{1st} & \text{breche?} & \text{brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \text{2nd} & \text{brichst} & \text{brecht} & \text{brichtest? brachst?} & \text{brichtet? bracht?} \\ \hline \text{3rd} & \text{bricht? brecht?} & \text{brichen? brechen?} & \text{brichte? brach?} & \text{brichten? brachen?} \\ \hline \multicolumn{2}{|c|}{\text{singular}} & \multicolumn{2}{c|}{\text{plural}} & \\ \hline \multicolumn{2}{|c|}{\text{present}} & \multicolumn{2}{c|}{\text{past}} & \\ \hline \end{array}\right)$$

3. **Text and paradigms**

$$p\left(\right)$$



Tokens

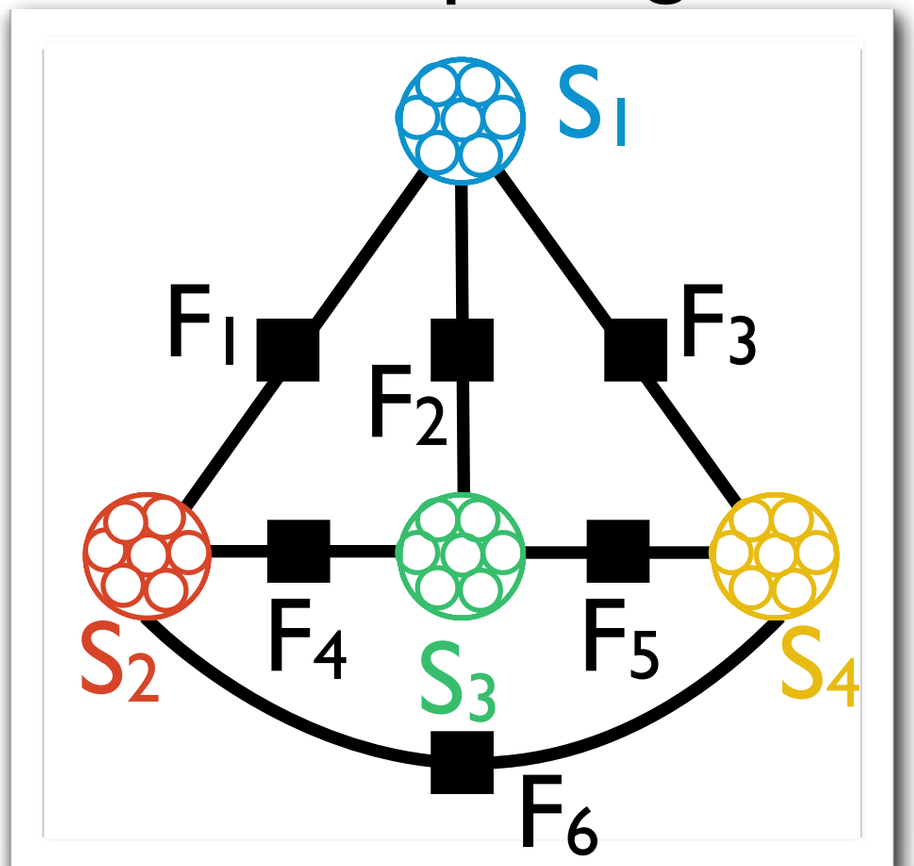


Types

3 Text & Paradigms

- We have seen how an **inflectional paradigm** ("multiple strings") can be modeled by **finite-state Markov Random Fields** (MRFs)
- Now we will build a joint model over **inflectional paradigms** and a **text corpus**
- **Goal:** Learn how to inflect words in the language, using clues from the text corpus

4 systematically related spellings:



Pr(S1, S2, S3, S4)

3 Text & Paradigms

Why do we want to use a text corpus?

- In **Part 2**:
  We learn how to inflect words, given some observed paradigms (complete or incomplete) that someone created as training data *(expensive supervision)* ☹
- Here in **Part 3**:
  We *also* want to learn how to inflect words, we'll use a few observed paradigms as well, but mainly learn from plain text (*cheap data*) ☺

3 Text & Paradigms

How can a text corpus help?

It can potentially **fix** erroneous MRF string predictions.

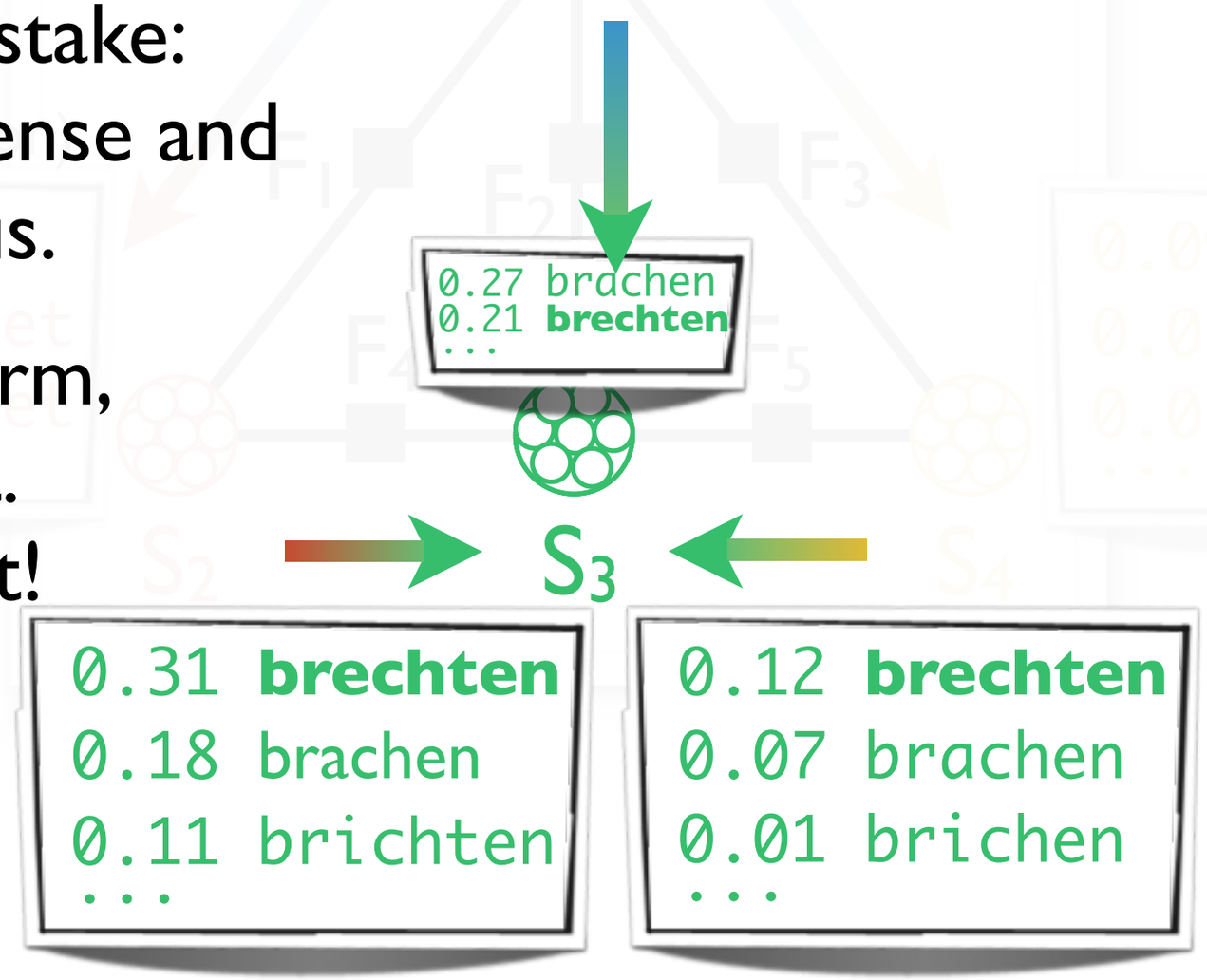
Intuition:
If a spelling predicted by the MRF **cannot be found in the corpus**, it was probably an incorrect prediction.

3 Text & Paradigms

Decoding output for S_3 (consensus): **brechten**

MRF is making a mistake:
brechten is nonsense and **not** found in corpus.

But the 2nd-best form, brachen, is frequent.
It's probably correct!



3 Text & Paradigms

We will make such decisions

using **statistical inference**,

under a **probability model** that uses the finite-state MRFs, but models the text corpus as well.

3 Text & Paradigms

What kind of probability model do we want?

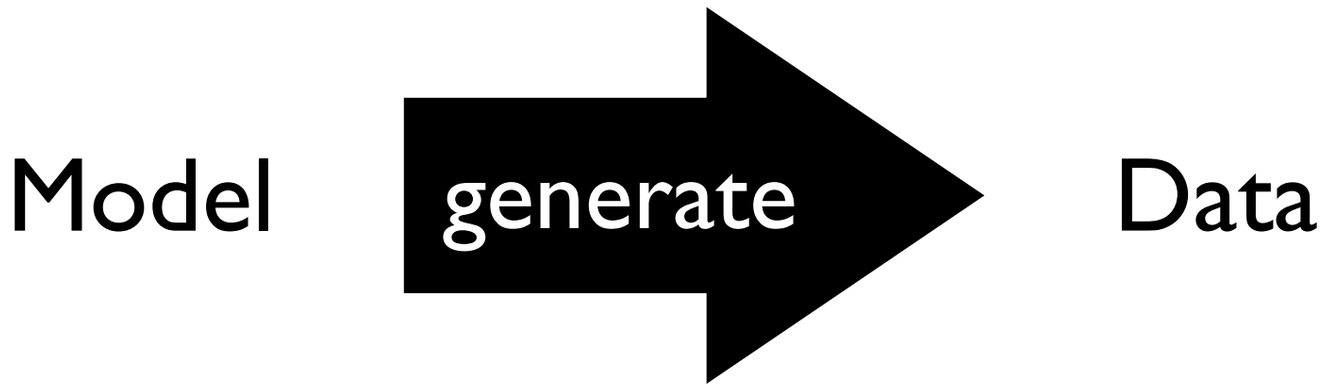
- Keep **tokens** and **types** separate
- Tokens are in the **text corpus**
- Types are in the **paradigms**
- We also model **abstract morphological knowledge** about how forms are related

3 Text & Paradigms

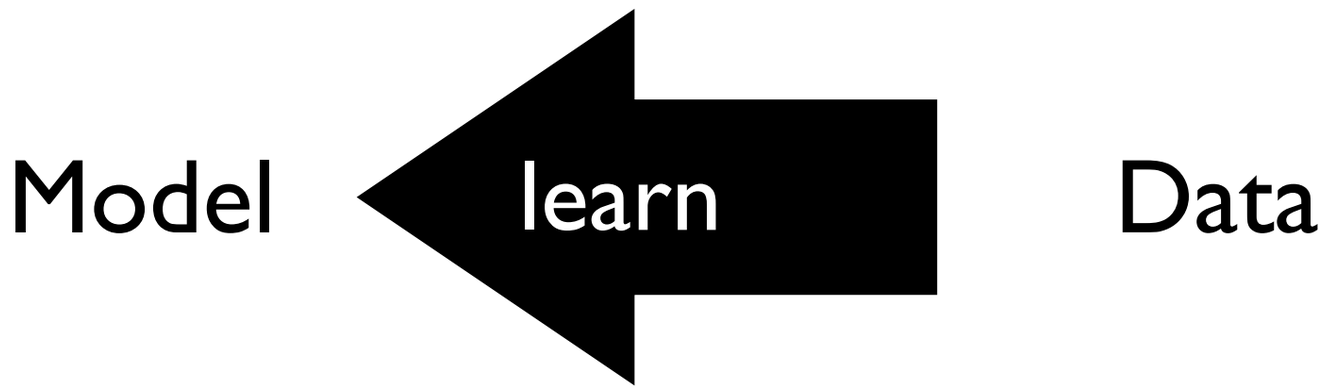
- Each paradigm contains the systematically related spellings of a **lexeme** (modeled by finite-state MRFs).
- Assume an **unbounded** number of possible lexemes and paradigms ("non-parametric") in the text.

3 Text & Paradigms

Generative story

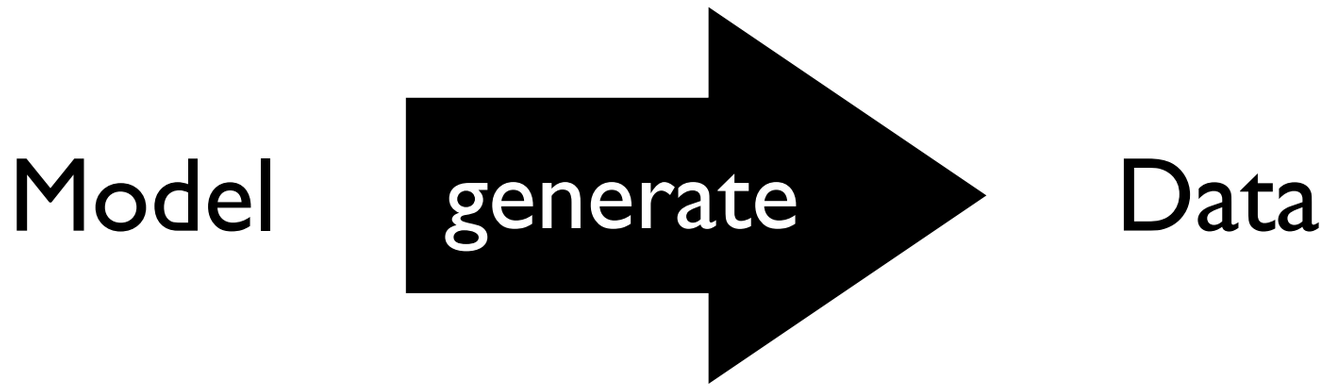


Inference (Sampling)



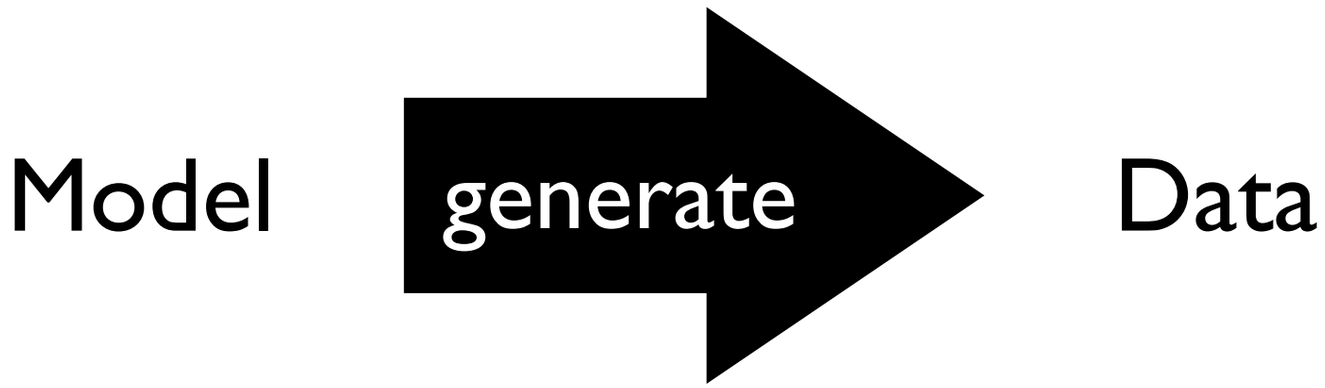
3 Text & Paradigms

Generative story



3 Text & Paradigms

Generative story



To generate from our model:
First, generate the **types** of the language.
Then, use them to generate the corpus **tokens**.

3 Text & Paradigms

(1) Generate infinitely many lexemes

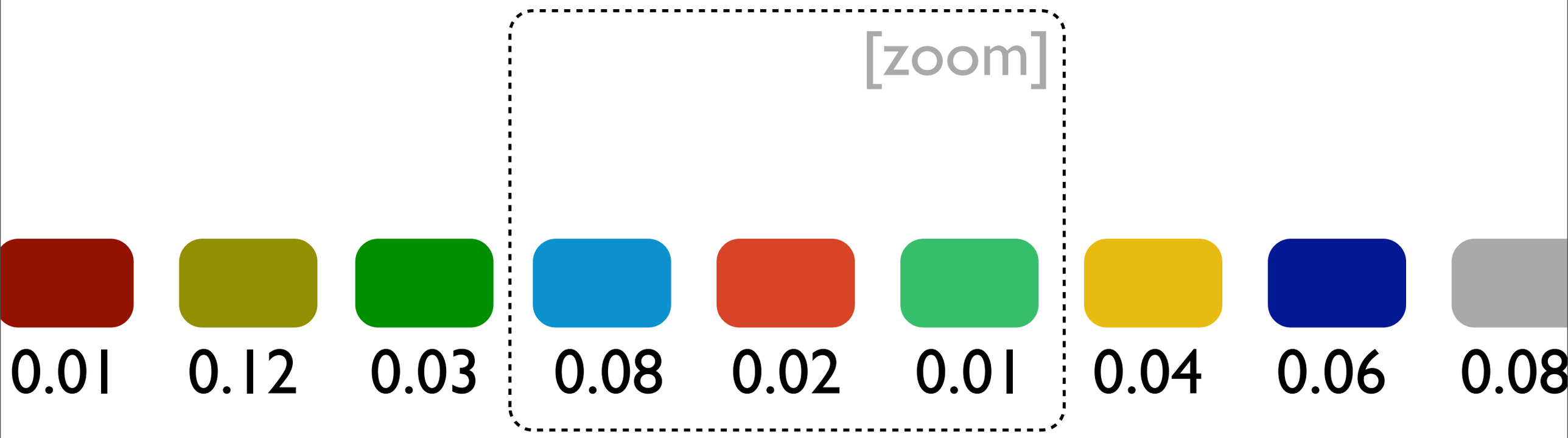
3 Text & Paradigms

(1) Generate infinitely many lexemes

Stick-breaking process

3 Text & Paradigms

(1) Generate infinitely many lexemes



Stick-breaking process

3 Text & Paradigms [zoom]

(1) Generate infinitely many lexemes

0.08 0.02 0.01

3 Text & Paradigms

(1) Generate infinitely many lexemes

0.08 0.02 0.01

3 Text & Paradigms

(2) Each lexeme has a paradigm with slots for the different inflections

1st sg	1st pl
2nd sg	2nd pl
3rd sg	3rd pl

0.08

0.02

0.01

3 Text & Paradigms

(2) Each lexeme has a paradigm with slots for the different inflections

1st sg	1st pl
2nd sg	2nd pl
3rd sg	3rd pl

0.08

1st sg	1st pl
2nd sg	2nd pl
3rd sg	3rd pl

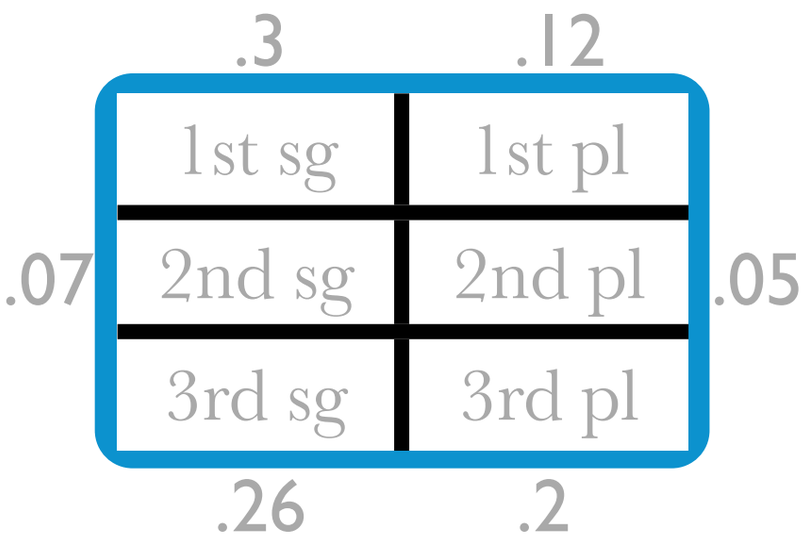
0.02

1st sg	1st pl
2nd sg	2nd pl
3rd sg	3rd pl

0.01

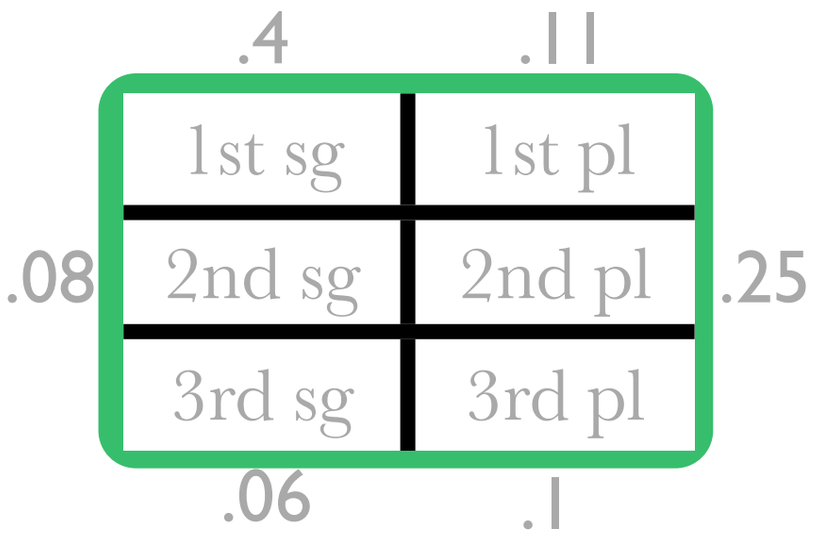
3 Text & Paradigms

(3) Each paradigm has a distribution over slot frequencies



0.08

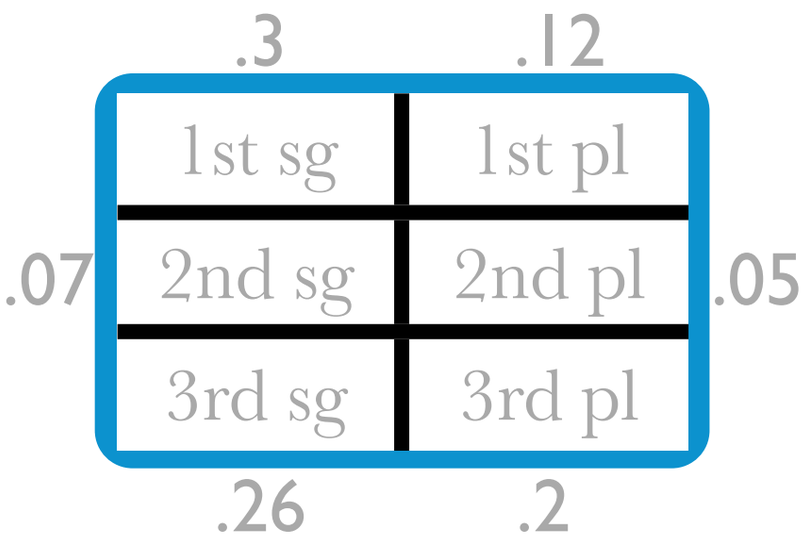
0.02



0.01

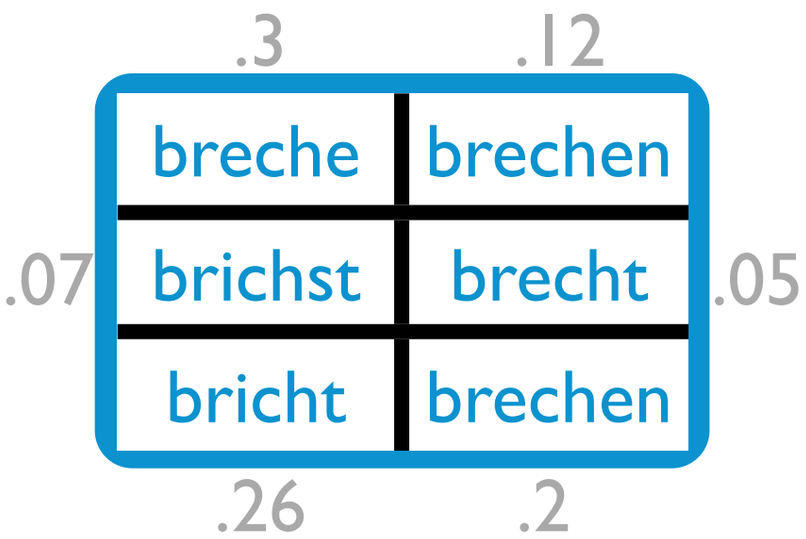
3 Text & Paradigms

(4) All paradigms generate their spellings using the same finite-state MRF parameterized by **θ**



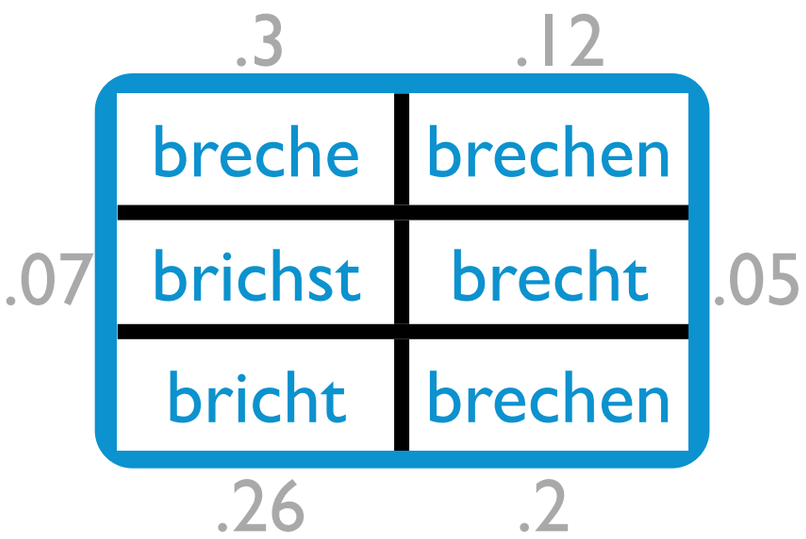
3 Text & Paradigms

(4) All paradigms generate their spellings using the same finite-state MRF parameterized by **θ**

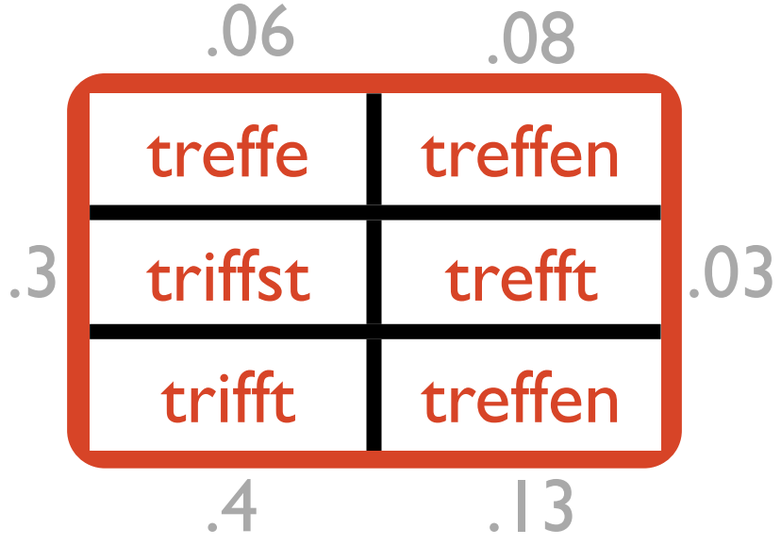


3 Text & Paradigms

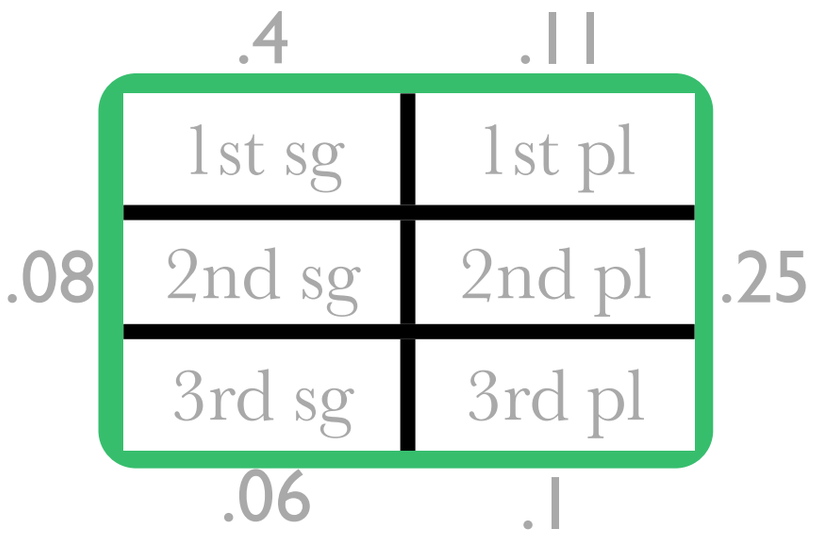
(4) All paradigms generate their spellings using the finite-state MRF parameterized by **θ**



0.08



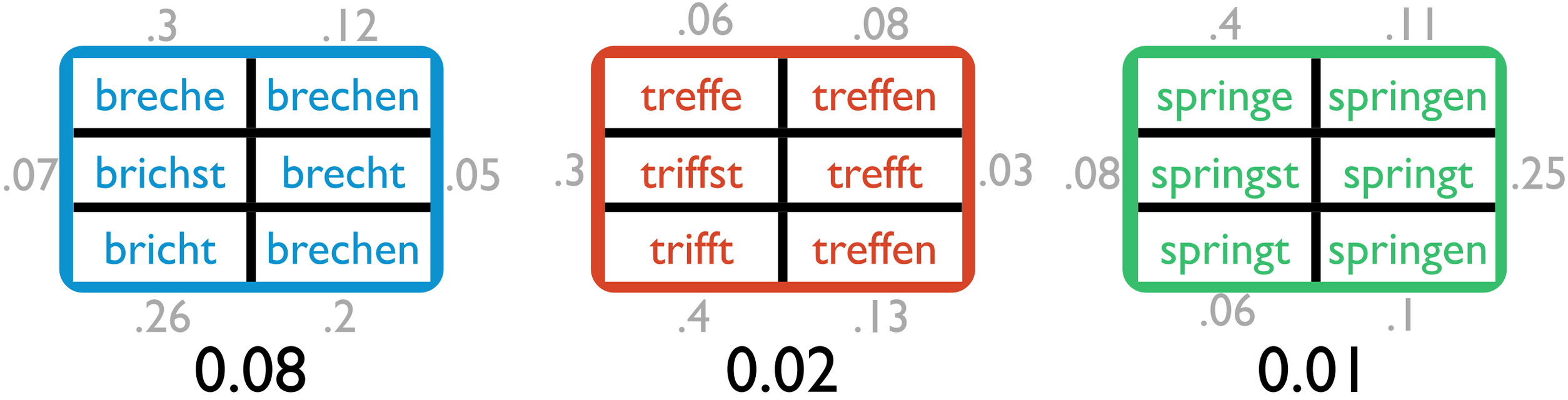
0.02



0.01

3 Text & Paradigms

(4) All paradigms generate their spellings using the finite-state MRF parameterized by **θ**



3 Text & Paradigms

(4) All paradigms generate their spellings using the finite-state MRF parameterized by **θ**

θ is a "morphological grammar"
The true θ for German morphology will know, for example,
"from *1st singular* to *plural*, just append the **suffix** *n*", or
"between *3rd sg* and *pl*, a **vowel change** is likely"

.3	.12
breche	brechen
brichst	brecht
bricht	brechen

.07 .05 .26 .2

0.08

.06	.08
treffe	treffen
triffst	trefft
trifft	treffen

.3 .03 .4 .13

0.02

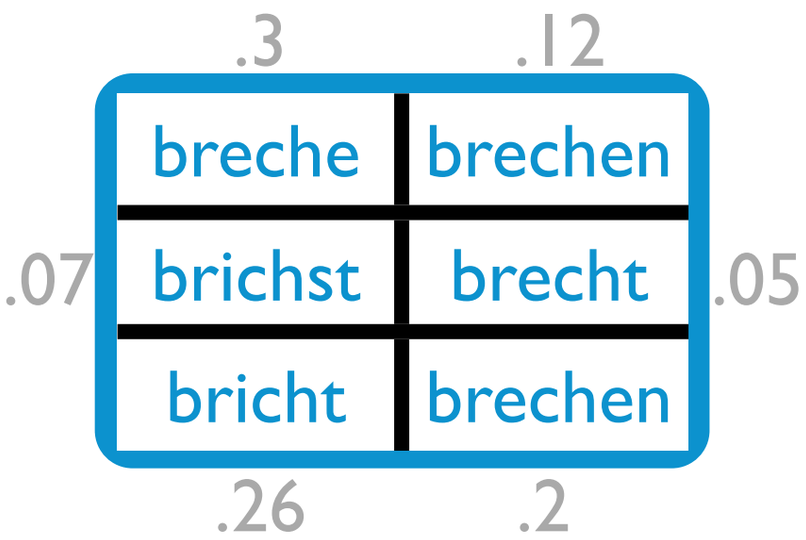
.4	.11
springe	springen
springst	springt
springt	springen

.08 .25 .06 .1

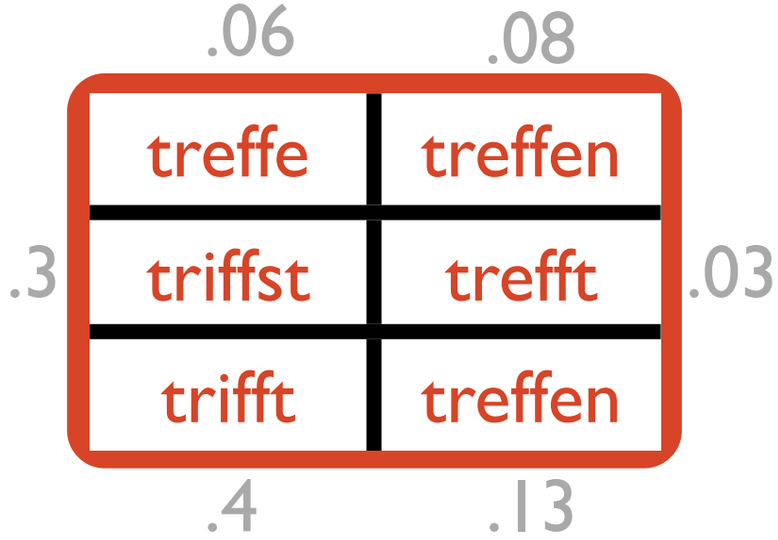
0.01

3 Text & Paradigms

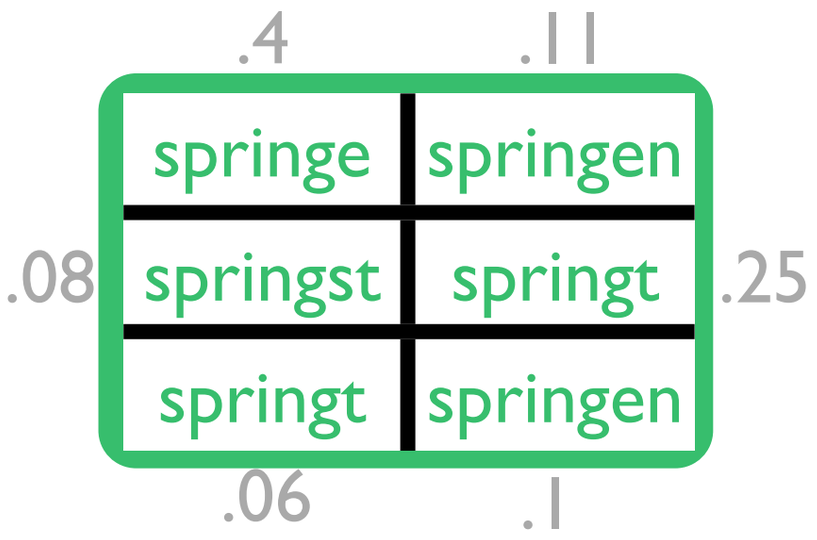
All types of the language have been generated.
Now generate the corpus tokens.



0.08

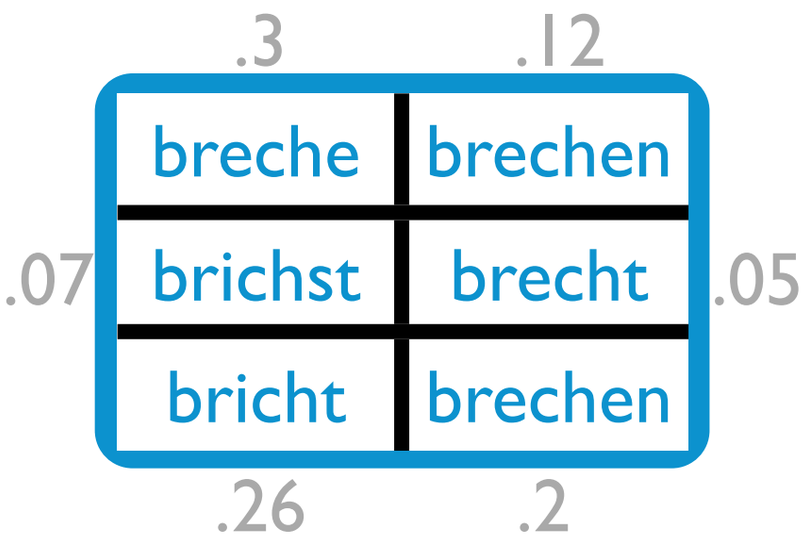


0.02

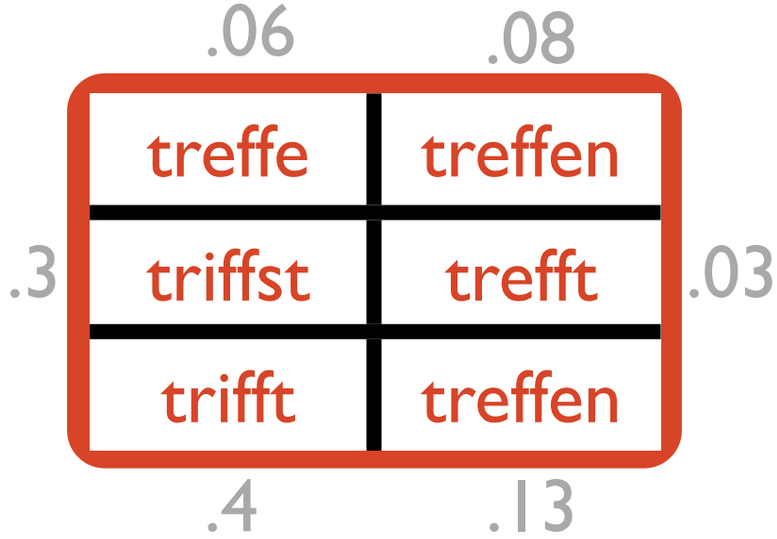


0.01

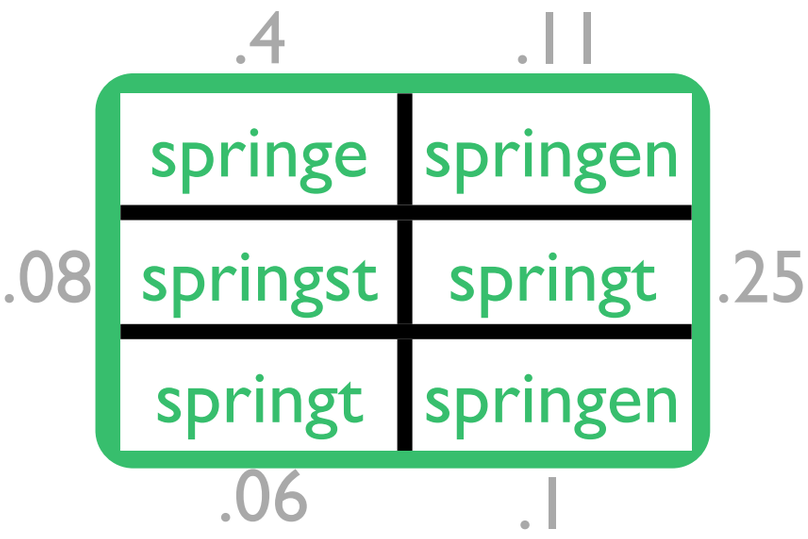
3 Text & Paradigms



0.08



0.02



0.01

3 Text & Paradigms

(5) Generate the corpus: *POS tags*

Adv V PPER V N V Prep V N V

.3	.12
breche	brechen
brichst	brecht
bricht	brechen

.07 .05 .26 .2

0.08

.06	.08
treffe	treffen
triffst	trefft
trifft	treffen

.3 .03 .4 .13

0.02

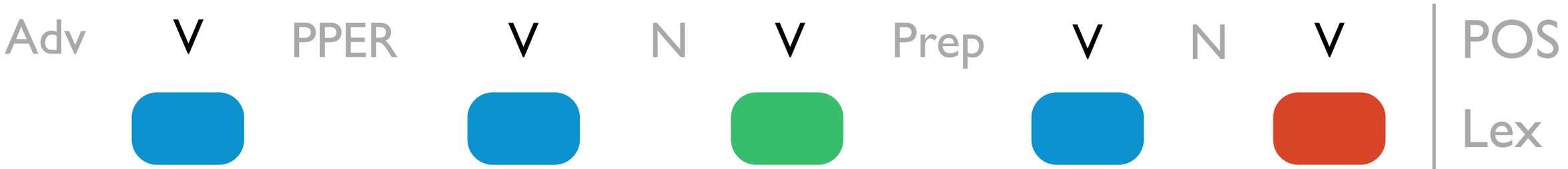
.4	.11
springe	springen
springst	springt
springt	springen

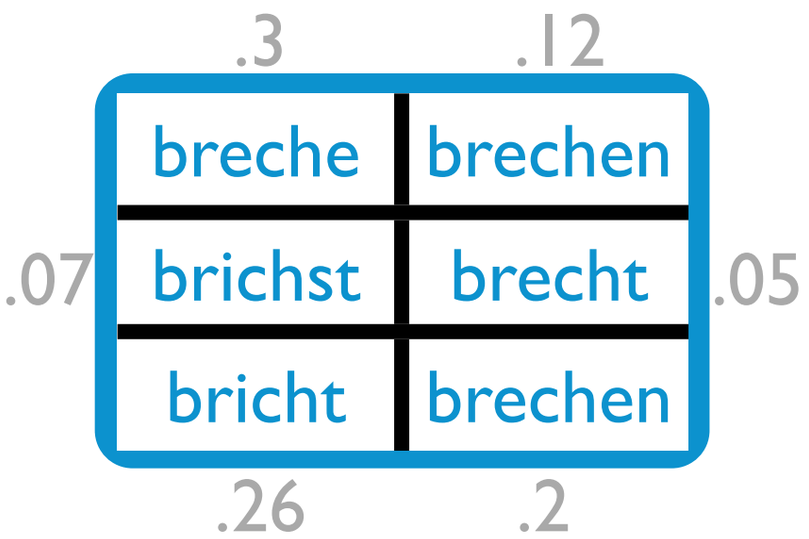
.08 .25 .06 .1

0.01

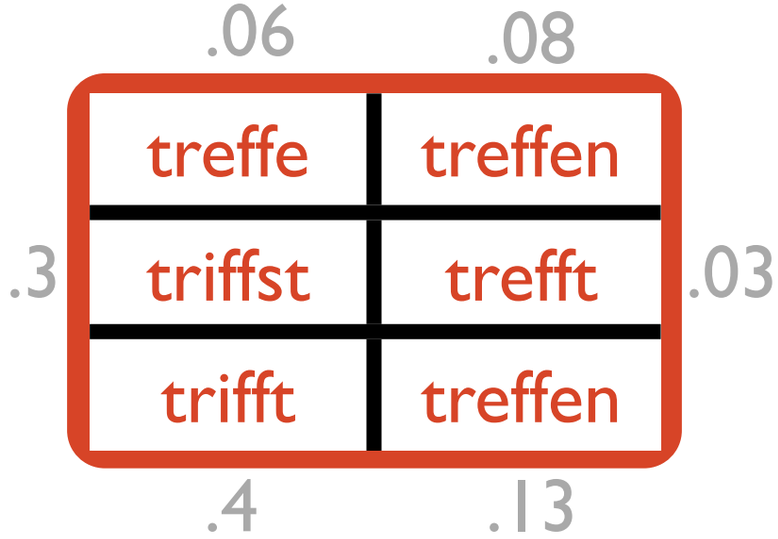
3 Text & Paradigms

(5) Generate the corpus: *Lexemes*

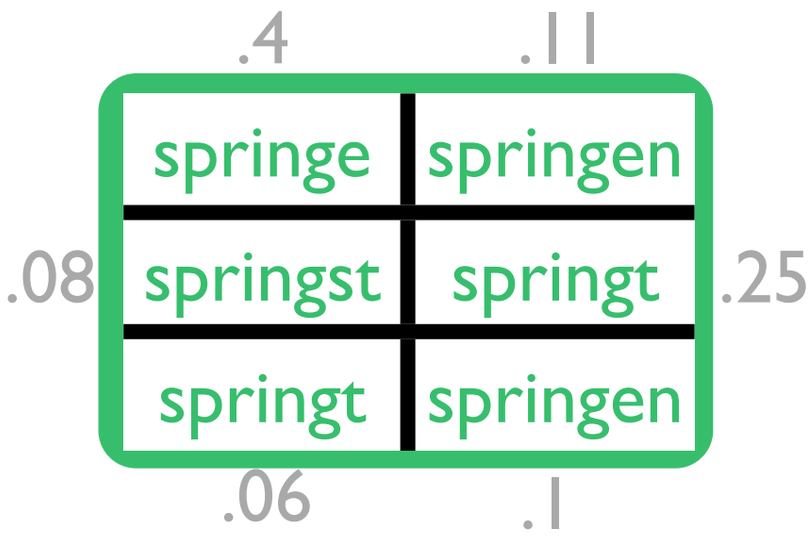




0.08



0.02



0.01

3 Text & Paradigms

(5) Generate the corpus: *Inflections*

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	3rd sg		1st pl		2nd pl		1st pl		2nd sg	Infl

.3	.12
breche	brechen
brichst	brecht
bricht	brechen

.07 .05 .26 .2

0.08

.06	.08
treffe	treffen
triffst	trefft
trifft	treffen

.3 .03 .4 .13

0.02

.4	.11
springe	springen
springst	springt
springt	springen

.08 .25 .06 .1

0.01

3 Text & Paradigms

(5) Generate the corpus: Look up the *spellings*

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	3rd sg		1st pl		2nd pl		1st pl		2nd sg	Infl
	bricht		brechen		springt		brechen		triffst	Spell

.3	.12
breche	brechen
brichst	brecht
bricht	brechen

.07 .05 .26 .2

0.08

.06	.08
treffe	treffen
triffst	trefft
trifft	treffen

.3 .03 .4 .13

0.02

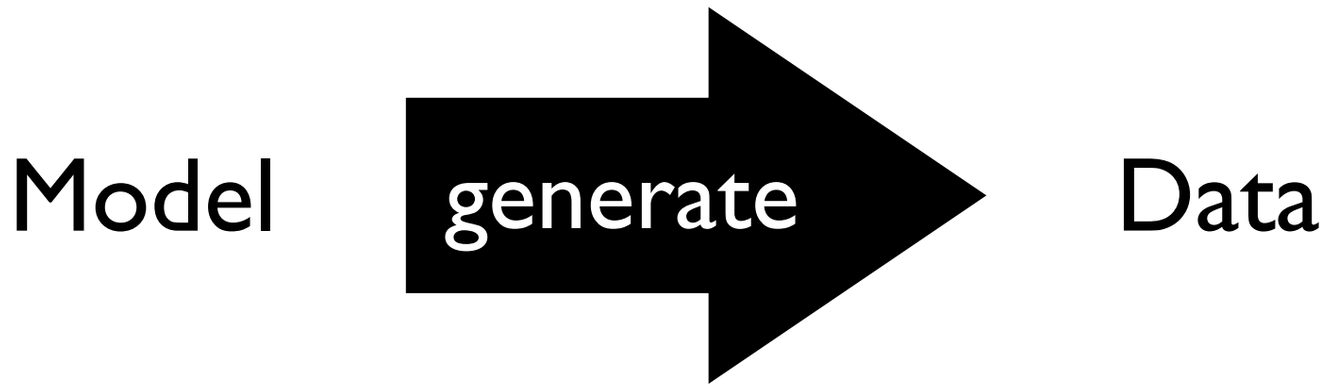
.4	.11
springe	springen
springst	springt
springt	springen

.08 .25 .06 .1

0.01

3 Text & Paradigms

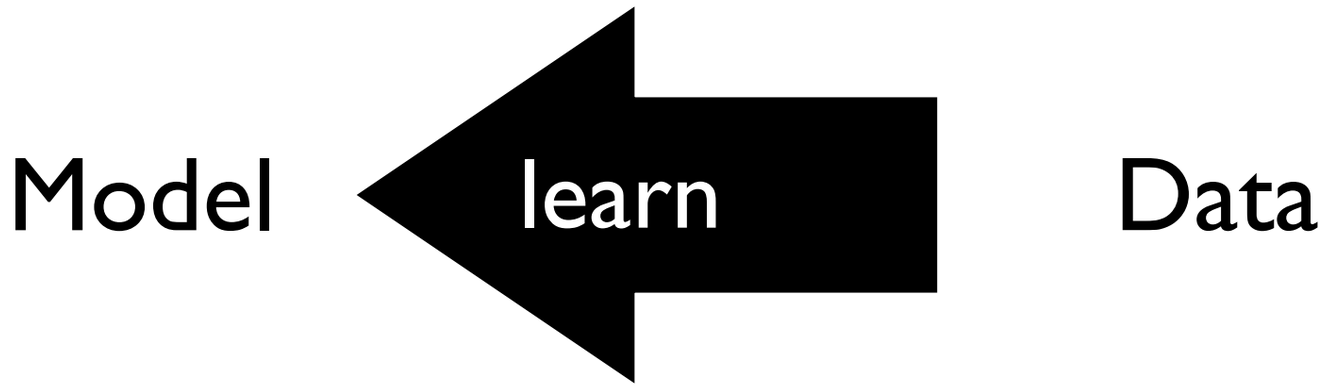
Generative story



3 Text & Paradigms

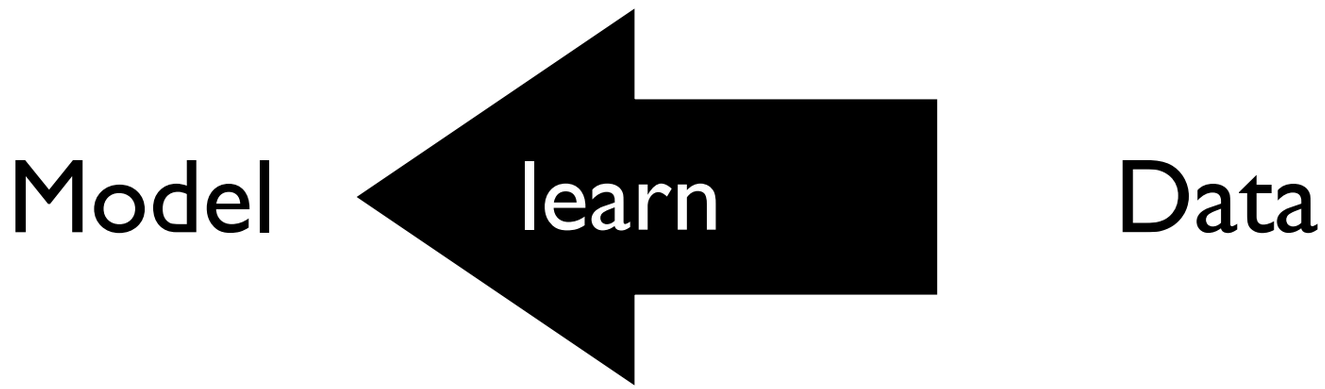
3 Text & Paradigms

Inference (Sampling)



3 Text & Paradigms

Inference (Sampling)



We start with **observed corpus tokens** and construct the paradigms and estimate all distributions

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	3rd sg		1st pl		2nd pl		1st pl		2nd sg	Infl
	bricht		brechen		springt		brechen		triffst	Spell

.07	breche	brechen	.05	.3
	brichst	brecht		
	bricht	brechen		
	.26	.2		

0.08

.3	treffe	treffen	.03	.08
	triffst	trefft		
	trifft	treffen		
	.4	.13		

0.02

.4	springe	springen	.25	.06
	springst	springt		
	springt	springen		
	.06	.1		

0.01

3 Text & Paradigms

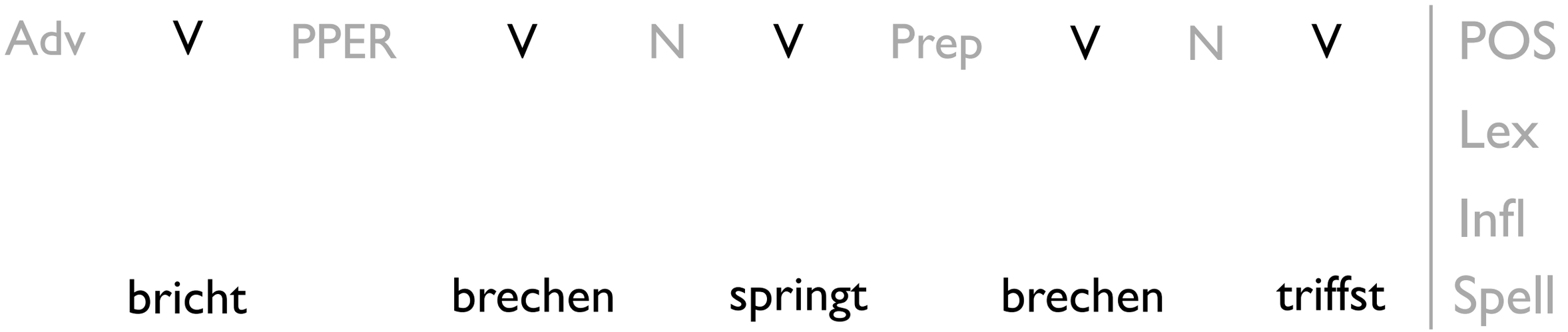
Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
bricht		brechen		springt		brechen		triffst		Infl
										Spell

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
bricht		brechen		springt		brechen		triffst	

POS
Lex
Infl
Spell

3 Text & Paradigms



Minimal supervision: We do also observe a **few seed paradigms**, from which we can estimate an initial θ, which parameterizes the finite-state MRFs

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
										Infl
bricht		brechen		springt		brechen		triffst		Spell

Seed paradigm

treffe	treffen
triffst	trefft
trifft	treffen

Minimal supervision: We do also observe a **few seed paradigms**, from which we can estimate an initial θ, which parameterizes the finite-state MRFs

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
										Infl
	bricht		brechen		springt		brechen		triffst	Spell

Seed paradigm

treffe	treffen
triffst	trefft
trifft	treffen

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
										Infl
bricht		brechen		springt		brechen		triffst		Spell

Seed paradigm

treffe	treffen
triffst	trefft
trifft	treffen

Train initial θ values

"morphological grammar"

"e" is likely to change into "i"

3rd sg ends in "t"

from 3rd sg to 1st pl, change vowel

...

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	bricht		brechen		springt		brechen		triffst

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
bricht		brechen		springt		brechen		triffst	
W1		W2		W3		W4		W5	

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

The red lexeme is completely specified and “bricht” does not fit in.

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
bricht W1		brechen W2		springt W3		brechen W4		triffst W5	

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

1st sg	1st pl
2nd sg	2nd pl
3rd sg	3rd pl

Remember: θ says,

3rd sg ends in “t”

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
		bricht W1		brechen W2		springt W3		brechen W4	triffst W5

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

	1st sg	1st pl
	2nd sg	2nd pl
W1	3rd sg	3rd pl

Remember: θ says,

3rd sg ends in “t”

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg								
	bricht		brechen		springt		brechen		triffst
	W1		W2		W3		W4		W5

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

1st sg		1st pl
2nd sg		2nd pl
W1 bricht		3rd pl

Remember: θ says,

3rd sg ends in “t”

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg								
	bricht		brechen		springt		brechen		triffst
	W1		W2		W3		W4		W5

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brichen? brechen?

We immediately run finite-state-based **belief propagation** in this new paradigm.

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg								
	bricht	**brechen**	springt	brechen	triffst				
	W1	W2	W3	W4	W5				

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? **brechen**?
brichst? brechst?	bricht? brecht?
W1 bricht	brichen? **brechen?**

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg		3rd pl						
	bricht		**brechen**	springt	brechen		triffst		
	W1		W2	W3	W4		W5		

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg		3rd pl						
	bricht		brechen	**springt**	brechen		triffst		
	W1		W2	W3	W4		W5		

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg		3rd pl						
	bricht		brechen		**springt**		brechen		triffst
	W1		W2		W3		W4		W5

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2

1st sg	1st pl
2nd sg	2nd pl
3rd sg	3rd pl

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg		3rd pl		3rd sg				
	bricht		brechen		**springt**		brechen		triffst
	W1		W2		W3		W4		W5

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2

1st sg	1st pl
2nd sg	2nd pl
W3 springt	3rd pl

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg		3rd pl		3rd sg				
	bricht		brechen		**springt**		brechen		triffst
	W1		W2		W3		W4		W5

POS
Lex
Infl
Spell

Run belief propagation!

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
W3 springt	springen? sprengen?

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg		3rd pl		3rd sg				
	bricht		brechen		springt		**brechen**		triffst
	W1		W2		W3		W4		W5

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
W3 springt	springen? sprengen?

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg		3rd pl		3rd sg				
	bricht		brechen		springt		**brechen**		triffst
	W1		W2		W3		W4		W5

POS
Lex
Infl
Spell

It would fit well in two of the cells:

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? **brechen**?
brichst? brechst?	bricht? brecht?
W1 bricht	**brechen** W2

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
W3 springt	springen? sprengen?

3 Text & Paradigms

Do not have to run BP now, because paradigm spellings are not changed.
But frequency estimates change right away.

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	3rd sg		3rd pl		3rd sg		3rd pl			Infl
	bricht		brechen		springt		**brechen**		triffst	Spell
	W1		W2		W3		W4		W5	

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2 W4

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
W3 springt	springen? sprengen?

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V
	3rd sg		3rd pl		3rd sg		3rd pl		
	bricht		brechen		springt		brechen		**triffst**
	W1		W2		W3		W4		W5

POS
Lex
Infl
Spell

treffe	treffen
triffst	trefft
trifft	treffen

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2 W4

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
W3 springt	springen? sprengen?

3 Text & Paradigms

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	3rd sg		3rd pl		3rd sg		3rd pl		2nd sg	Infl
	bricht		brechen		springt		brechen		**triffst**	Spell
	W1		W2		W3		W4		W5	

treffe	treffen
triffst	trefft
trifft	treffen

W5

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2 W4

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
W3 springt	springen? sprengen?

3 Text & Paradigms

We will now re-estimate θ, given our new "observations" (samples). This training method is called MCEM.

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	3rd sg		3rd pl		3rd sg		3rd pl		2nd sg	Infl
	bricht		brechen		springt		brechen		**triffst**	Spell
	W1		W2		W3		W4		W5	

treffe	treffen
triffst	trefft
trifft	treffen

W5

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2 W4

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
W3 springt	springen? sprengen?

3 Text & Paradigms

We go over the corpus over and over again, **re-analyzing words** in the light of **newly acquired knowledge** about table frequencies, inflection frequencies and the updated "morphological grammar" θ.

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	3rd sg		3rd pl		3rd sg		3rd pl		2nd sg	Infl
	bricht		brechen		springt		brechen		**triffst**	Spell
	W1		W2		W3		W4		W5	

treffe	treffen
triffst	trefft
trifft	treffen

W5

briche? breche?	brichen? brechen?
brichst? brechst?	bricht? brecht?
W1 bricht	brechen W2 W4

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
W3 springt	springen? sprengen?

3 Text & Paradigms

We go over the corpus over and over again, **re-analyzing words** in the light of **newly acquired knowledge** about table frequencies, inflection frequencies and the updated "morphological grammar" θ.

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	3rd sg		3rd pl		3rd sg		3rd pl		2nd sg	Infl
	bricht		brechen		springt		brechen		triffst	Spell
	W1		W2		W3		W4		W5	

treffe	treffen
triffst	trefft
trifft	treffen

W5

briche? breche?	brichen? brechen?
brichst? brechst?	**bricht?** brecht?
bricht	brechen

W1 W2 W4

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
springt	springen? sprengen?

W3

3 Text & Paradigms

We go over the corpus over and over again, **re-analyzing words** in the light of **newly acquired knowledge** about table frequencies, inflection frequencies and the updated "morphological grammar" θ.

Adv	V	PPER	V	N	V	Prep	V	N	V	POS
										Lex
	2nd pl		3rd pl		3rd sg		3rd pl		2nd sg	Infl
	bricht		brechen		springt		brechen		triffst	Spell
	W1		W2		W3		W4		W5	

treffe	treffen
triffst	trefft
trifft	treffen

W5

briche? breche?	brichen? brechen?
brichst? brechst?	**bricht**
bricht? brecht?	brechen

W1 W2 W4

springe? sprenge?	springen? sprengen?
springst? sprengst?	springt? sprengt?
springt	springen? sprengen?

W3

3 Text & Paradigms

Summary of the sampling process:

- Constantly update **frequency estimates** for lexemes and inflections
- Often update the "morphological grammar" θ
- Keep **re-analyzing words** accordingly
- Run **finite-state BP** to fill in missing paradigm cells
- **Important:** Often, BP will produce a regular and some more irregular candidates, one of them is found in the corpus and placed in the cell, so we "learn" it!

3 Text & Paradigms

Summary of the sampling process:

- Inflections and lexemes at the corpus positions are **sampled**.
- The missing paradigm cells are **marginalized** over.
- The "morphological grammar" θ is **maximized**.
- We are using a collapsed **Gibbs** sampler, according to a hierarchical **Chinese Restaurant** Process, with interspersed finite-state **belief propagation** steps

3 Text & Paradigms

Sampling Speedup:

- Improve mixing and prevent "lock-in"
- Do not sample word by word
- Instead: Pick a whole lexeme, remove all its current words and perform Gibbs sampling just with those words (for one or more iteratons)

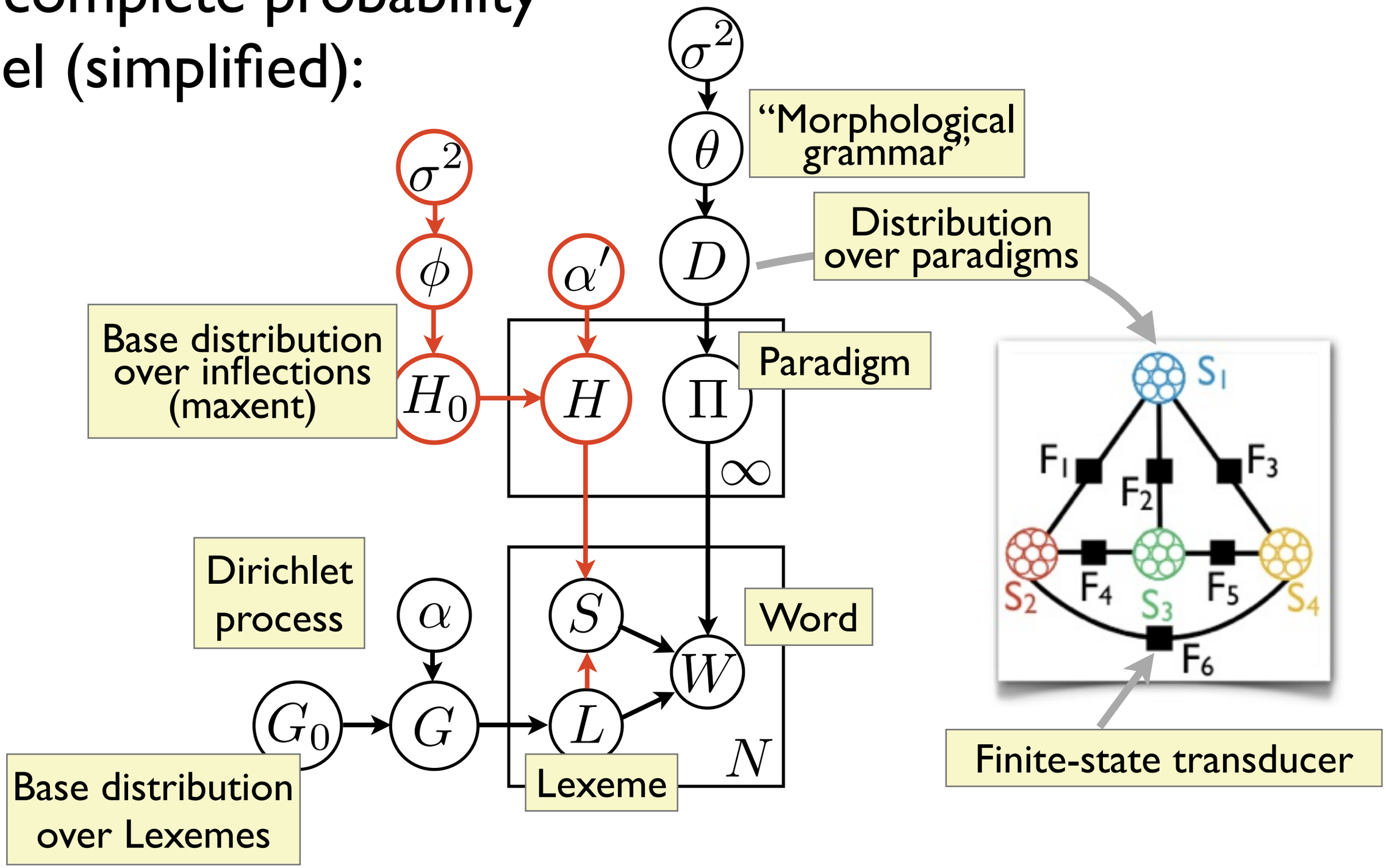
3 Text & Paradigms

Obtaining results for evaluation

- We add many paradigms, in which **only the lemma form** is given, but the other slots are empty.
- Just **keep track** of what corpus tokens the sampler places in those empty cells, or what candidates will be suggested from belief propagation.
- To get an answer for particular cell, get its **marginal** probability distribution at end of each iteration. At the end, get **average prob. per spelling** and report highest-scoring one

3 Text & Paradigms

The complete probability model (simplified):



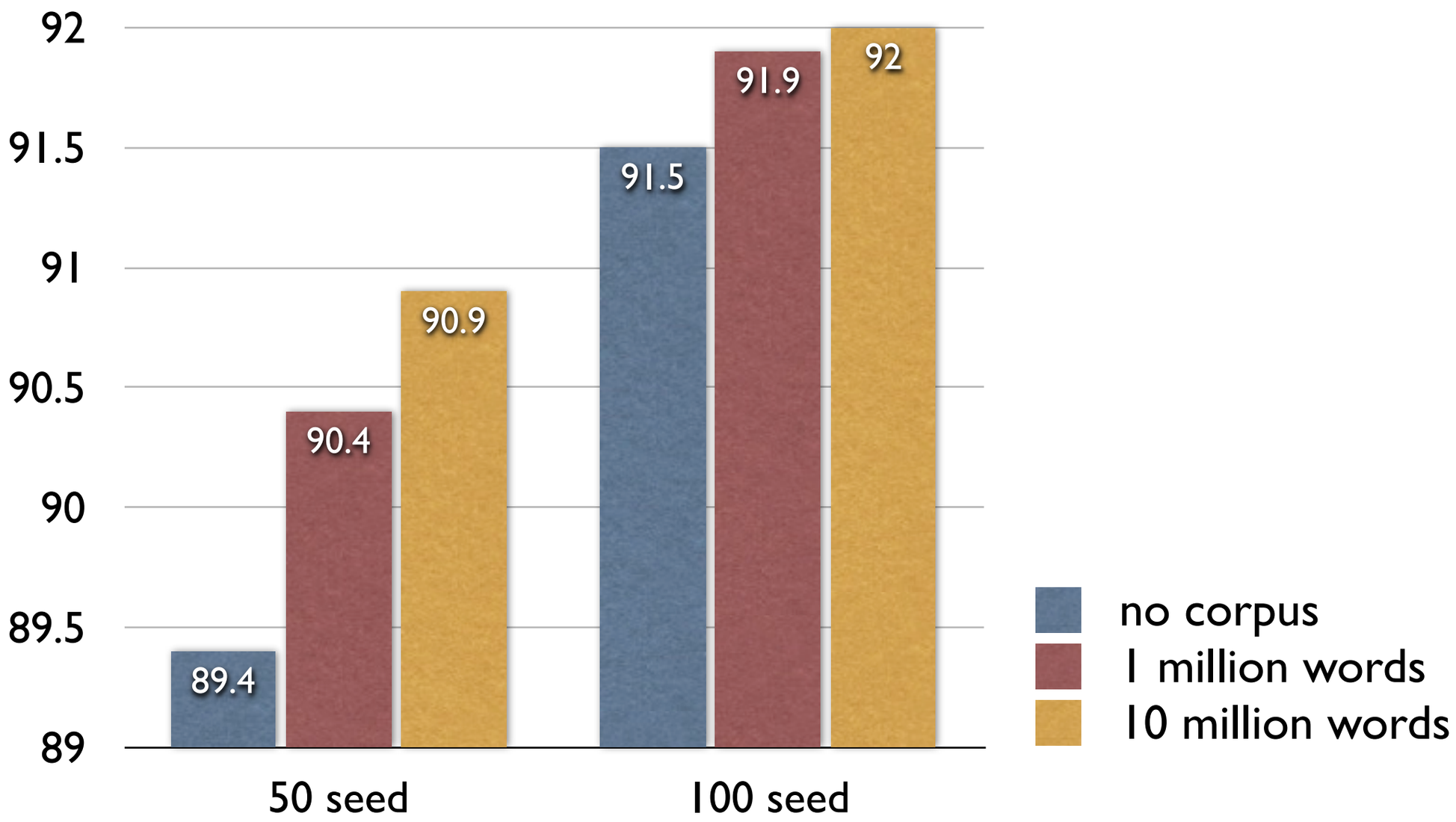
3 Text & Paradigms

Experiment:
Learn German inflectional morphology

- Given:
  - 50 seed paradigms (from CELEX)
  - German corpus of 10 million words (from "WaCKy" corpus)
- Test:
  For 5,415 German verbs, predict paradigms with 21 inflections each

3 Text & Paradigms

Adding a large text corpus significantly improves prediction accuracy.

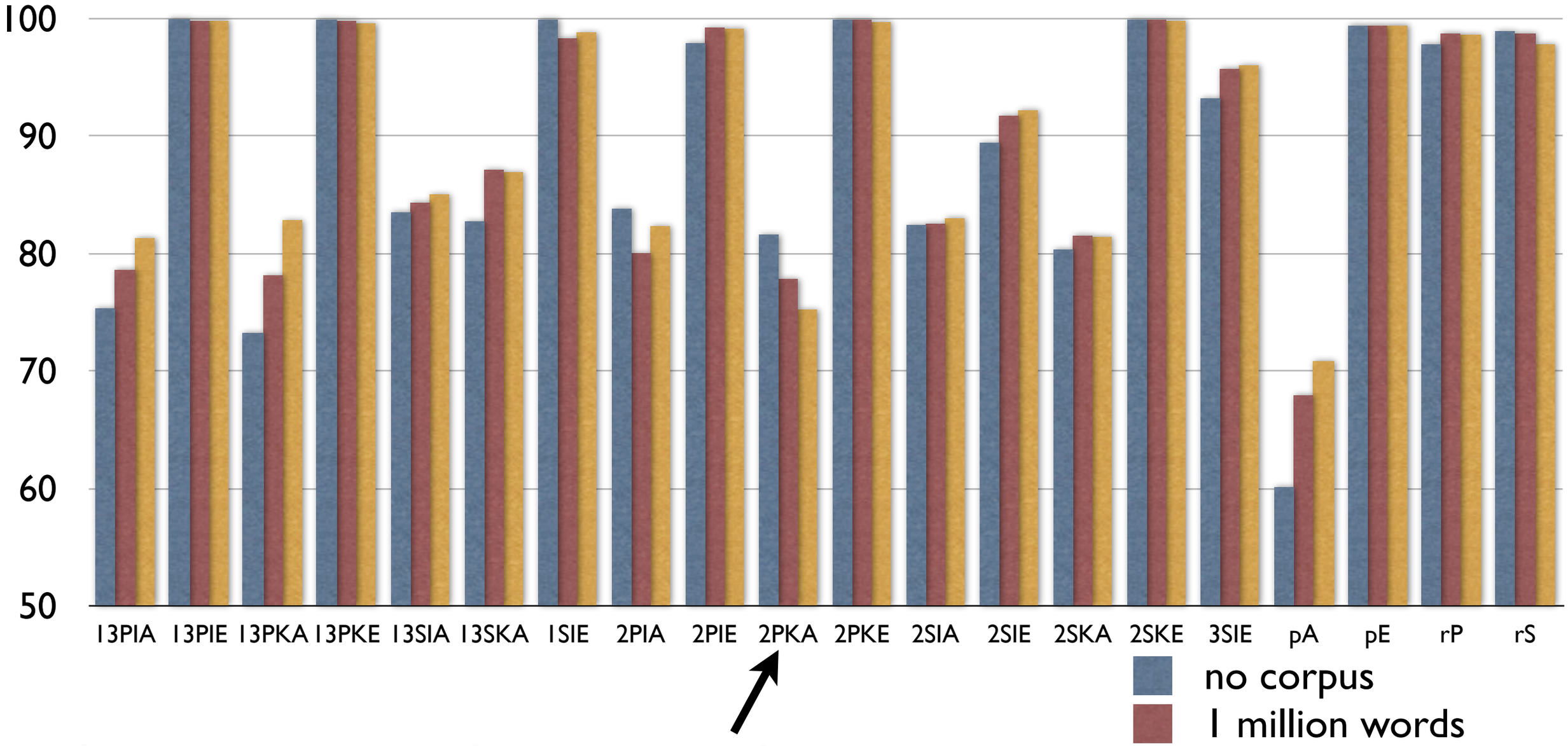


Regular: 85.4

3 Text & Paradigms

Many forms are easy (~100% acc.)

Large gains on some forms (irregularities)



In rare cases, the corpus hurts.

3 Text & Paradigms

- Simplifications:
  - The finite-state MRFs use a simple factor graph that just connects the lemma to all other forms, but not the forms among each other
  - Information flows from one form to the other through the lemma slot
  - Better factor graphs give orthogonal improvements, see Ch. 3

Text & Paradigms

Possible model extensions

- **Adding context:** Take neighboring words into account, so that *1st pl* can be more likely after "we" than after "she", etc.
- **Adding topic variables:** Useful for deciding that a particular spelling belongs into one lexeme rather than another (singed does not fit into "sing" paradigm because it's a different topic).

3 Text & Paradigms

Remaining morphological issues

- Reduplication
- Metathesis
- Consonant doubling
- Circumfixes
- Templatic morphology
- Interaction with phonology

3 Conclusions / Contributions

- Presented novel, principled approach to **learning inflectional morphology** of a language
- Developed joint **probability model** over **text corpus** and **inflectional paradigms**, which are hidden
- Presented **type-based sampling procedure** that discovers inflectional paradigms from plain text

3 Conclusions / Contributions

- Presented novel **generative story** for inflectional morphology, ...
- ... based on **ordinary linguistic notions** (lexemes, inflections, paradigms)
- Clusters corpus words into lexemes and inflections using hierarchical **Dirichlet process**
- Allows **unbounded number of lexemes**
- Handles **nonconcatenative**, irregular morphology

Related Work

1 String pairs

- *Sherif & Kondrak (2007)*, *Hong et al (2009)*, and others, get 1-best alignment, segment into chunks, and score chunks individually
- Others get 1-best alignment and train conventional *n*-gram model (*Bisani & Ney (2008)*, and others)
- In contrast, we sum over all alignments, use features, add latent variables, generate arbitrary-length output, use global normalization

Related Work

2 Multiple Strings

- Joint models over multiple strings have not been tackled much before
- Exception: *Bouchard-Cote et al (2007)*, who defines a directed graphical model, does not run BP inference and does not use FSTs

Related Work

3 Text and Paradigms

- No one has modeled structured inflectional paradigms before
- Typically, simple concatenative morphology is assumed (*Harris (1955), Chan (2008)*), but see *Yarowsky and Wicentowski (2002)*
- *Goldsmith (2001)* and others extract "suffix paradigms" (lists of verb endings)
- In contrast, we extract structured paradigms that seemlessly handle non-concatenative phenomena

Conclusions / Contributions

- Presented several novel probability models step by step, each smaller one being a factor component in the next bigger one
- Developed a coherent, unified statistical approach to inflectional morphology, which advances the state-of-the-art in computational morphology
- Extracted detailed and structured morphological knowledge from plain text;
- Presented the most ambitious morphological knowledge discovery task and method to date

Conclusions / Contributions

- All presented models have many further uses in NLP:
  - string-pair models for transliteration, pronunciation modeling, spelling correction, etc.
  - multiple-string models for bioinformatics, historical linguistics, phonology, transliteration, etc.
  - text & paradigms model for text generation, machine translation, etc.

Conclusions / Contributions

- This thesis naturally brings together many different concepts from machine learning, NLP, and linguistics, in various novel ways:
  - In part 1, we use linguistically inspired features, latent variables, FSTs and dynamic programming.
  - In part 2, we combine FSTs with graphical models and belief propagation.
  - In part 3, we bring together all of the above with statistical tools like Dirichlet process and collapsed Gibbs sampling to tackle a novel task that people have not been able to tackle before.

Publications

1. Dreyer and Eisner. *in prep*. Discovering Morphological Paradigms From Plain Text.
2. Dreyer and Eisner. 2009. Graphical Models over Multiple Strings. *EMNLP*.
3. McNamee, Dredze, Gerber, Garera, Finin, Mayfield, Piatko, Rao, Yarowsky, Dreyer. 2009. HLCOE Approaches to Knowledge Base Population. *TAC*.
4. Dreyer, Eisner and Smith. 2008. Finite-State Modeling of Log-Linear String Transductions with Latent Variables and Backoff Features. *EMNLP*.
5. Karakos, Eisner, Khudanpur, Dreyer. 2008. Machine Translation System Combination using ITG-based Alignments. *ACL*.
6. Dreyer and Shafran. 2007. Exploiting Prosody for PCFGs with Latent Annotations. *Interspeech*.
7. Dreyer, Hall, Khudanpur. 2007. Comparing Reordering Constraints for SMT Using Efficient BLEU Oracle Computation. *SSST*.
8. Dreyer and Eisner. 2006. Better Informed Training of Latent Syntactic Features. *EMNLP*.
9. Dreyer, Smith, Smith. 2006. Vine Parsing and Minimum Risk Reranking for Speed and Precision. *CoNLL*.
10. Burbank, Carpuat, Clark, Dreyer, Fox, Groves, Hall, Hearne, Melamed, Shen, Way, Wellington, Wu. 2005. Statistical Machine Translation by Parsing. *CLSP*.