Translation with Weighted Finite-State Devices

Some Models of Translation

- IBM Models 1-5
- Hidden Markov Model
- Phrase-Based Models

Q: What do all of these things have in common?

A: They all define *weighted regular languages* over a set of output sentences.

Desiderata

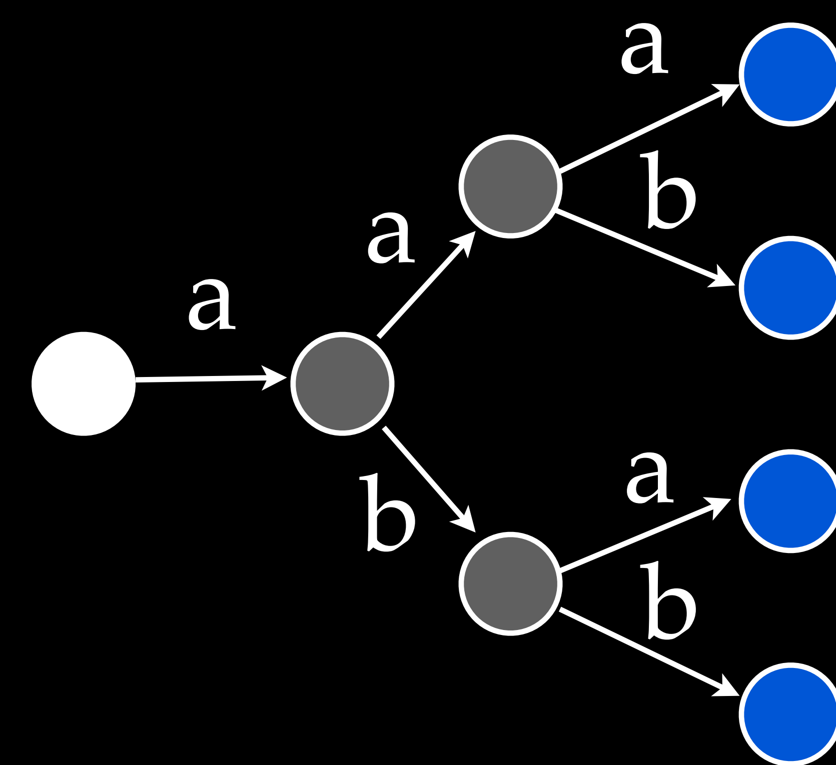
- We need efficient algorithms and data structures to:
  - Encode all of the strings in the language.
  - Assign probabilities to all of those strings.
    - Via products such as $p(e)p(f \mid e)$.
  - Find the string with the highest probability.
  - Compute expectations over substrings.
  - Compute mappings between strings.

Practical Implementation

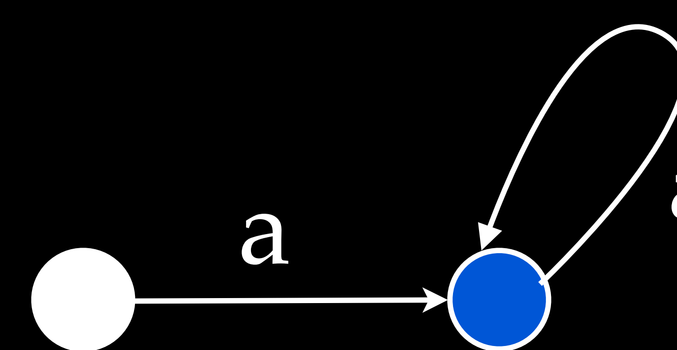
- Build each step as an individual transducer.
  - Compose at runtime (pruning at each step).
  - Assign probabilities to all of those strings.
    - Via products such as $p(e)p(f|e)$.
  - Find the string with the highest probability.
  - Compute expectations over substrings.
  - Compute mappings between strings.

Regular Languages

$$\mathcal{L}_1 = \left\{ \begin{array}{c} a\ a\ a \\ a\ b\ a \\ a\ a\ b \\ a\ b\ b \end{array} \right\}$$

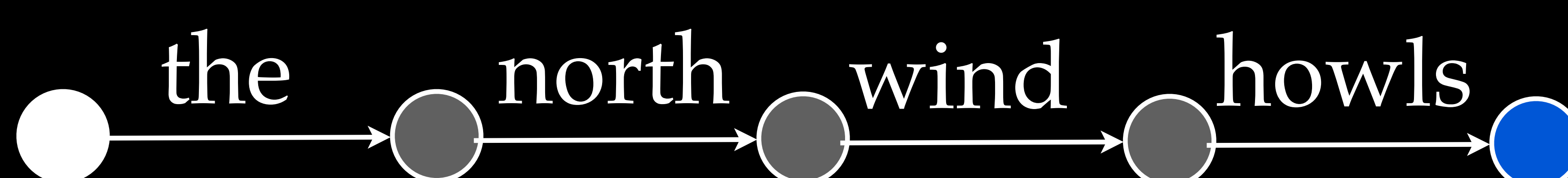


$$\mathcal{L}_2 = a^* = \{a, aa, aaa, \ldots\}$$



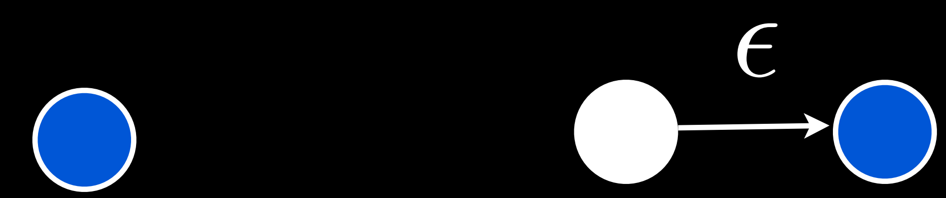
$\mathcal{L}_3$ = { “the north wind howls” }

finite-state automata

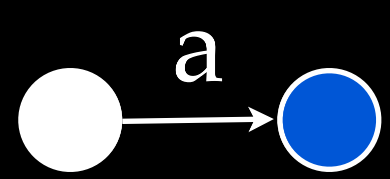


Regular Languages

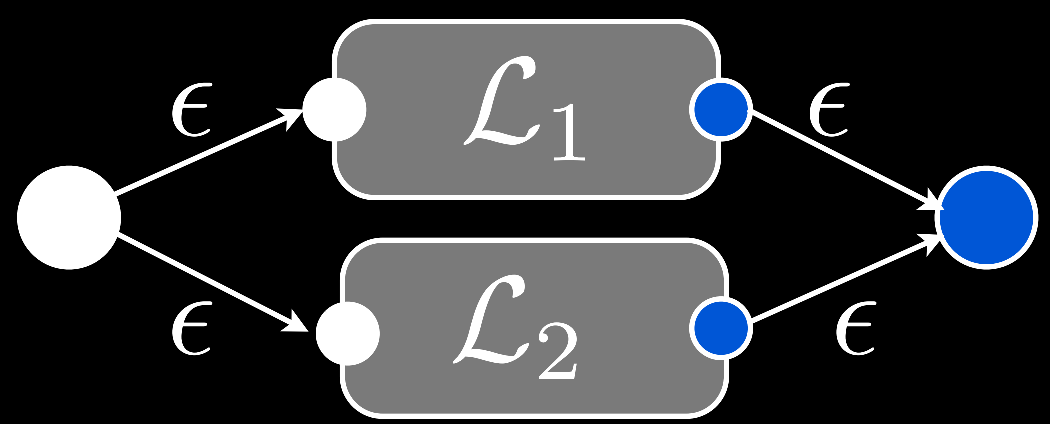
$\{\epsilon\}$ is regular



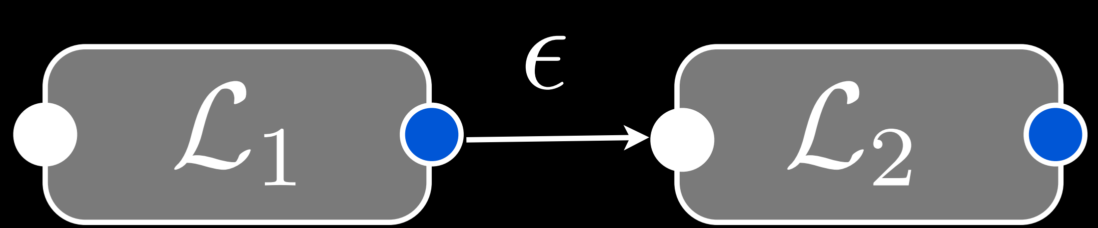
$\{a\}$ is regular



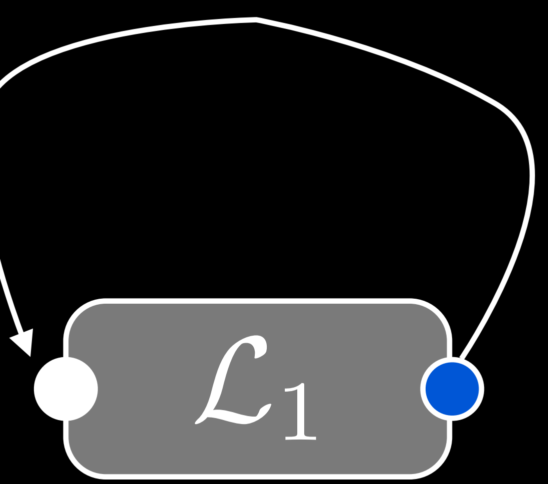
$\mathcal{L}_1 \cup \mathcal{L}_2$ is regular if $\mathcal{L}_1$ and $\mathcal{L}_2$ are regular



$\mathcal{L}_1 \cdot \mathcal{L}_2$



$\mathcal{L}_1^*$



Probabilistic Regular Languages

We want a function:

$$f : \mathcal{L} \to \mathbb{R}^+$$

such that:

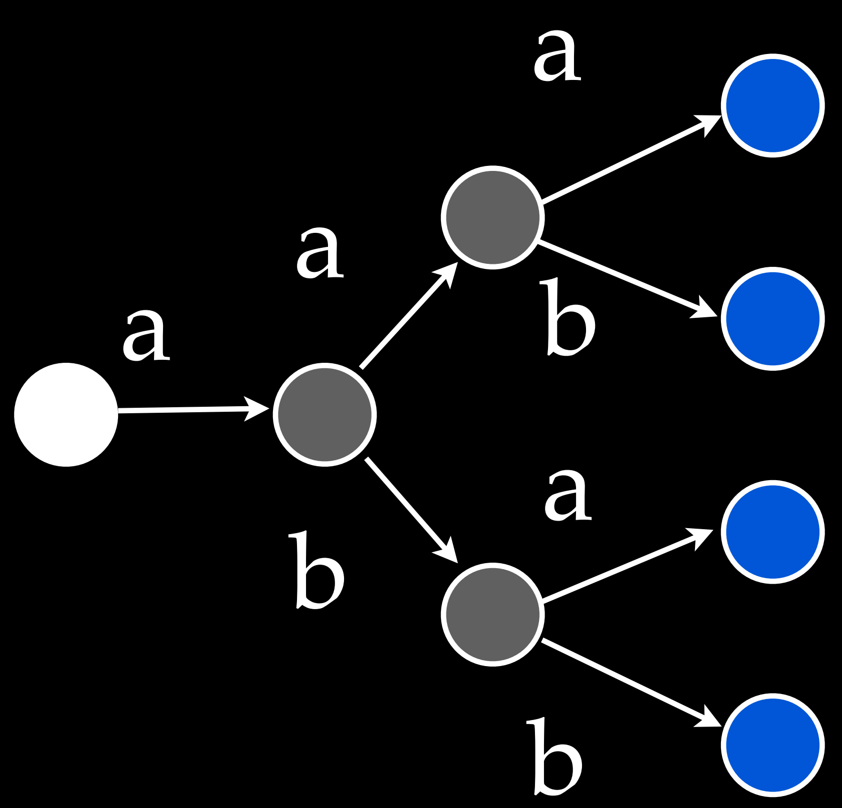
$$f(w) \in [0, 1]$$

$$\sum_w f(w) \in [0, 1]$$

Probabilistic Regular Languages

We want a function:

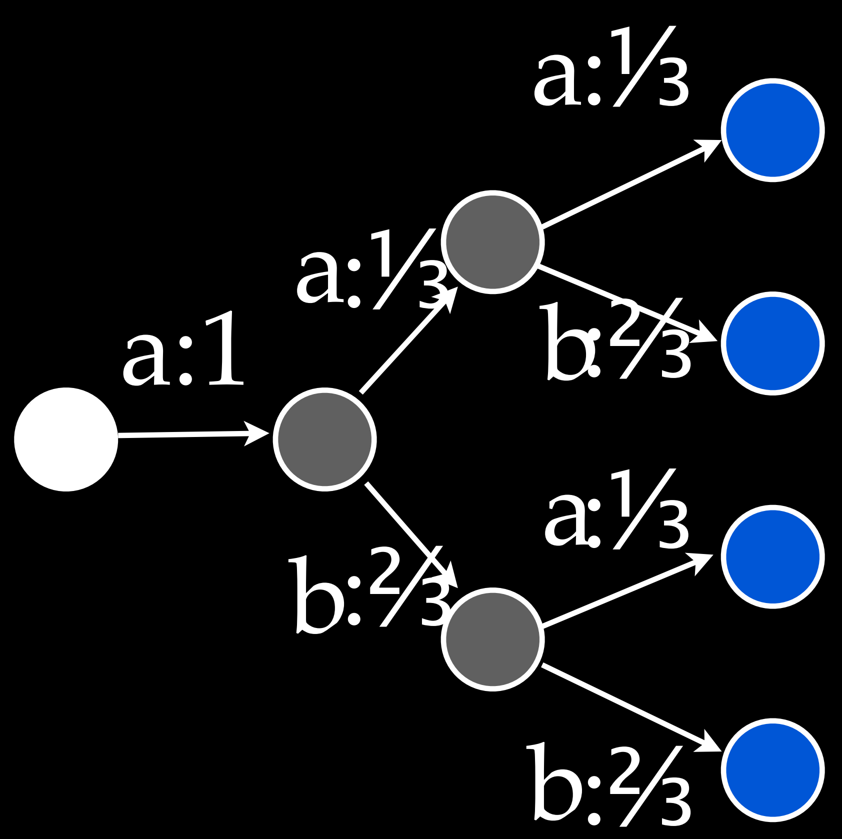
$$f : \mathcal{L} \rightarrow \mathbb{R}^{+}$$



Probabilistic Regular Languages

We want a function:

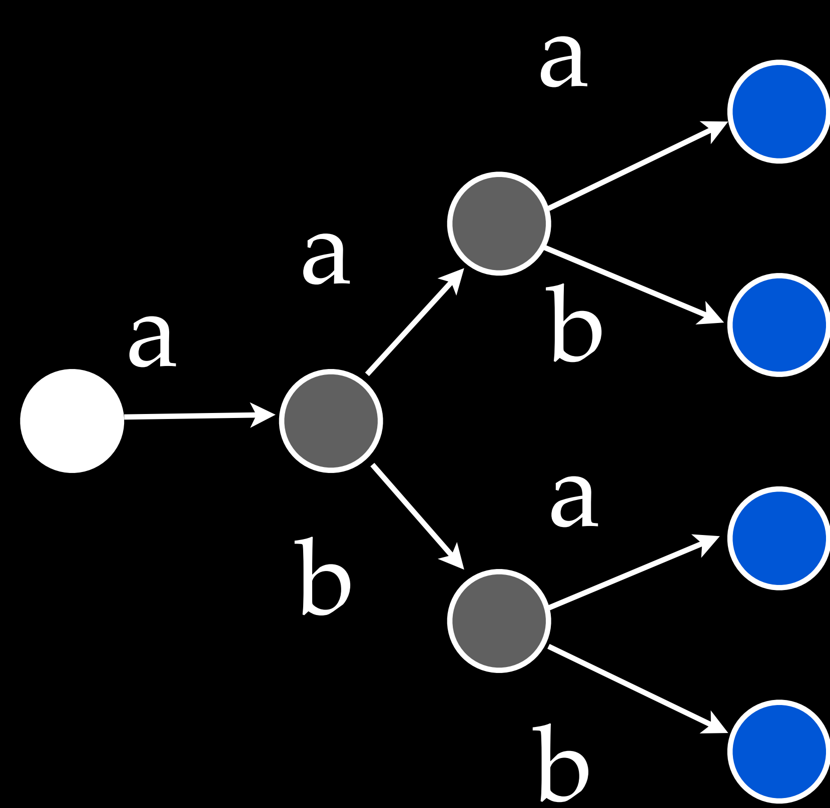
$$f : \mathcal{L} \rightarrow \mathbb{R}^+$$



Finite-State Transducers

We want a *binary relation*:

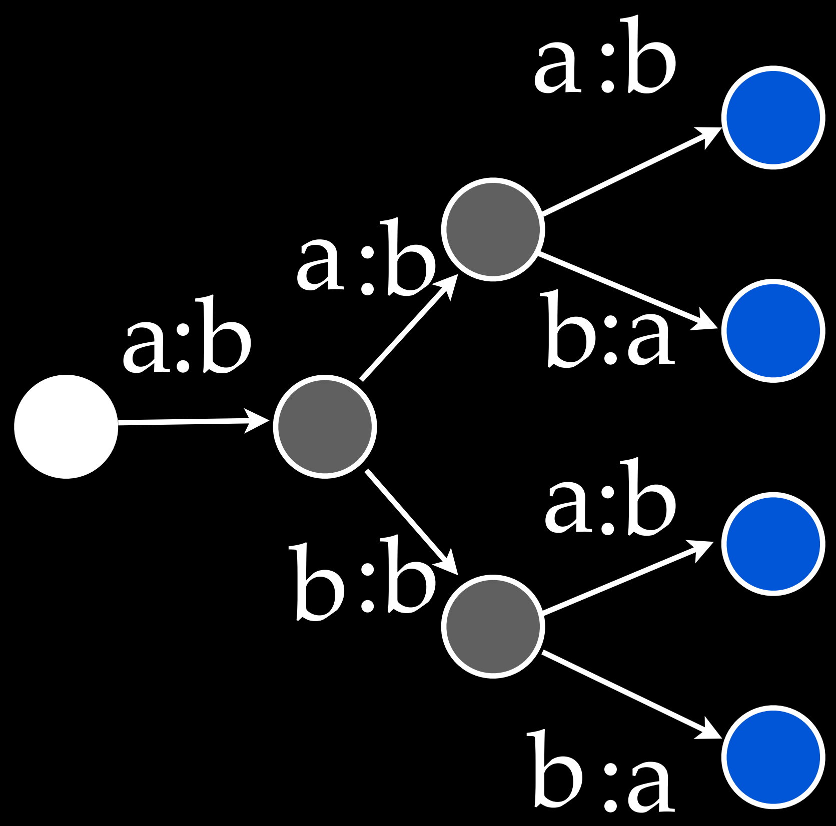
$$r \subseteq \mathcal{L}_1 \times \mathcal{L}_2$$



Finite-State Transducers

We want a *binary relation*:

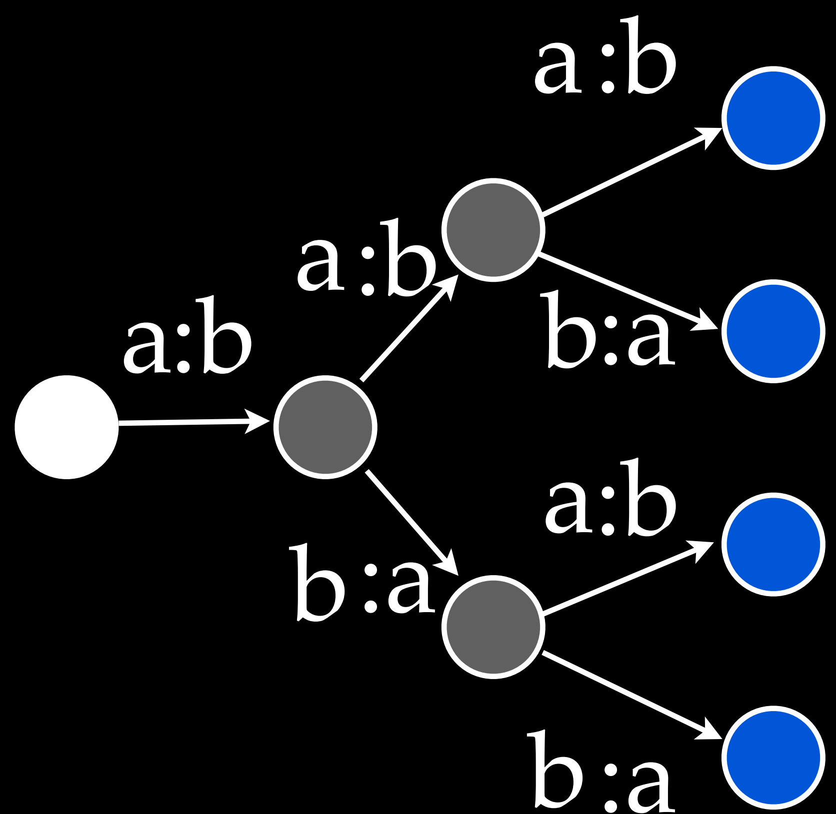
$$r \subseteq \mathcal{L}_1 \times \mathcal{L}_2$$



Finite-State Transducers

We want a *binary relation*:

$$r \subseteq \mathcal{L}_1 \times \mathcal{L}_2$$

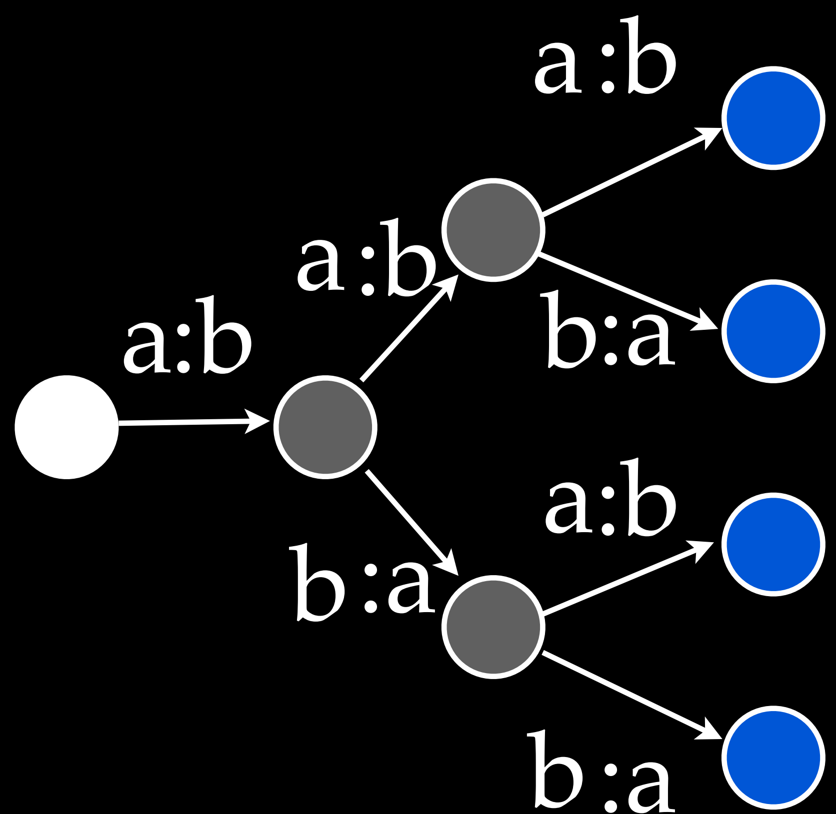


$$r = \left\{ \begin{array}{l} \text{a a a, b b b} \\ \text{a b a, b a b} \\ \text{a a b, b b a} \\ \text{a b b, b a a} \end{array} \right\}$$

Weighted Finite-State Transducers

We have a *binary relation*:

$$r \subseteq \mathcal{L}_1 \times \mathcal{L}_2$$



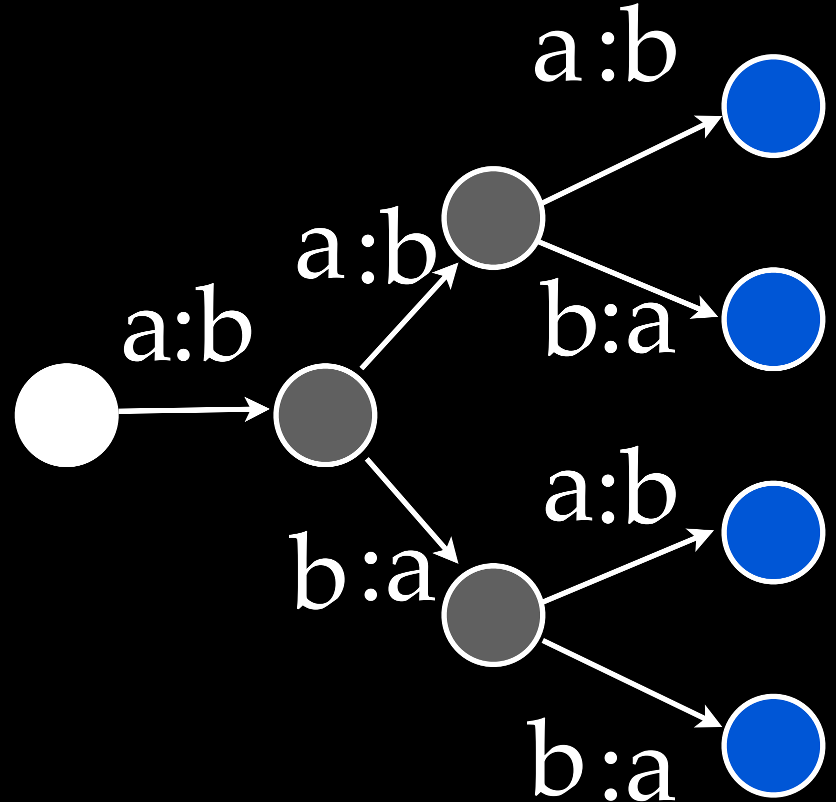
$$r = \left\{ \begin{array}{l} \text{a a a, b b b} \\ \text{a b a, b a b} \\ \text{a a b, b b a} \\ \text{a b b, b a a} \end{array} \right\}$$

Weighted Finite-State Transducers

We have a *binary relation*:

$$r \subseteq \mathcal{L}_1 \times \mathcal{L}_2$$

We want a function: $f : r \rightarrow \mathbb{R}^+$



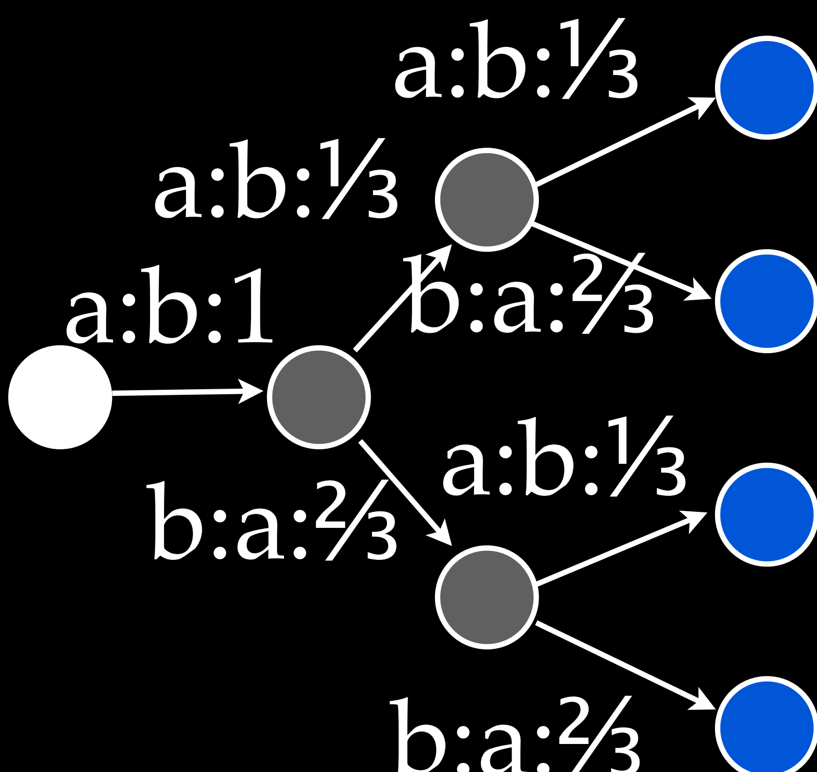
$$r = \left\{ \begin{array}{l} \text{a a a, b b b} \\ \text{a b a, b a b} \\ \text{a a b, b b a} \\ \text{a b b, b a a} \end{array} \right\}$$

Weighted Finite-State Transducers

We have a *binary relation*:

$$r \subseteq \mathcal{L}_1 \times \mathcal{L}_2$$

We have a function: $f : r \rightarrow \mathbb{R}^+$



$\langle$ a a a, b b b $\rangle \rightarrow$ 1⁄9

$\langle$ a b a, b a b $\rangle \rightarrow$ 2⁄9

$\langle$ a a b, b b a $\rangle \rightarrow$ 2⁄9

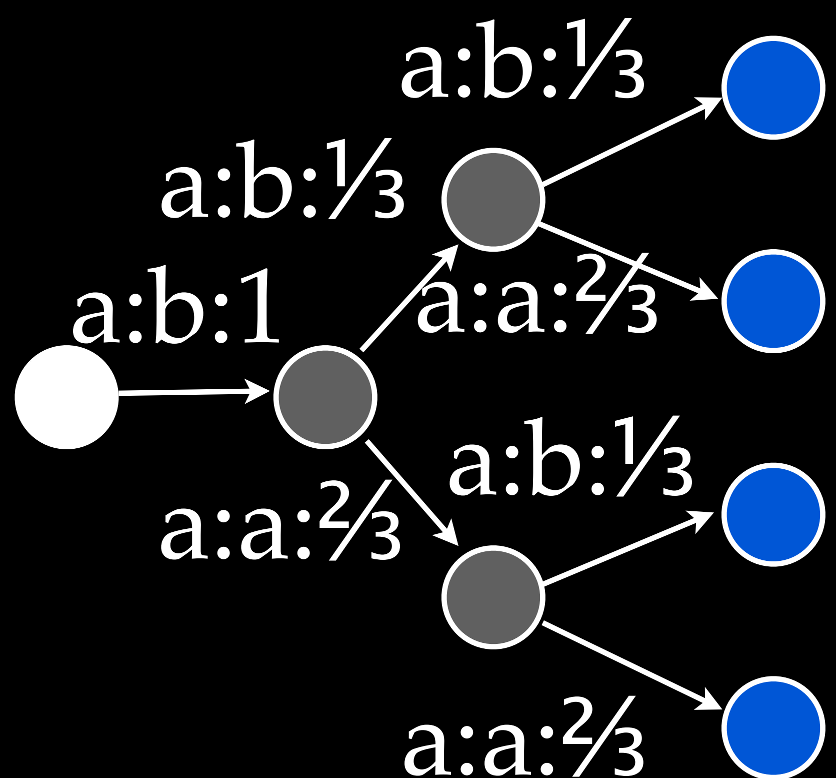
$\langle$ a b b, b a a $\rangle \rightarrow$ 4⁄9

Weighted Finite-State Transducers

We have a *binary relation*:

$$r \subseteq \mathcal{L}_1 \times \mathcal{L}_2$$

We have a function: $f : r \rightarrow \mathbb{R}^+$



$\langle$ a a a, b b b $\rangle \rightarrow$ 1/9

$\langle$ a a a, b a b $\rangle \rightarrow$ 2/9

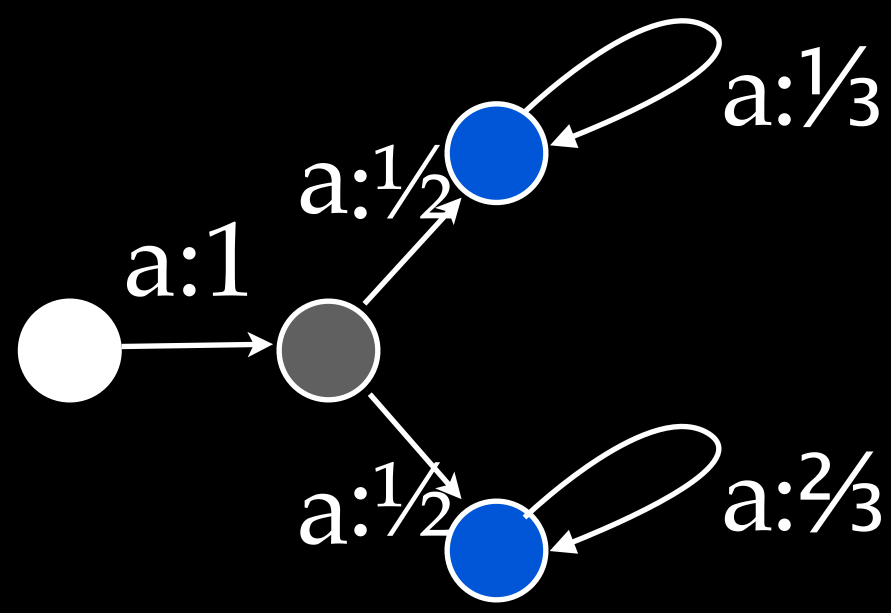
$\langle$ a a a, b b a $\rangle \rightarrow$ 2/9

$\langle$ a a a, b a a $\rangle \rightarrow$ 4/9

(Single string, multiple elements of *r*)

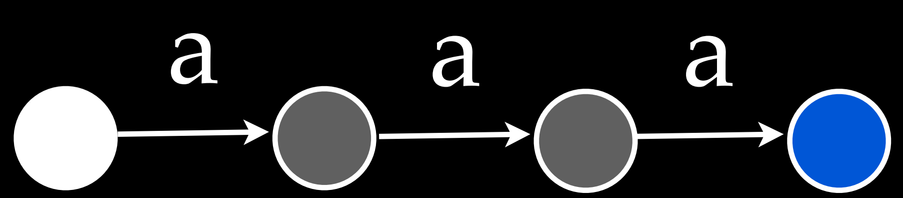
Algorithms don't Change...

- Shortest path (e.g. Dijkstra, A*): most probable *pair*
- Determinization (not all can be determinized)
  - But not w.r.t. to pairs, not single string!
- Lazy composition (e.g. intersection): $p(e)p(f|e)$

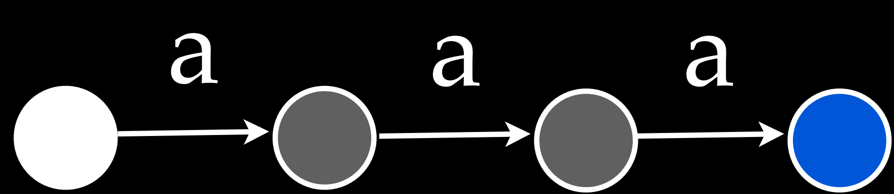


Composing Transducers

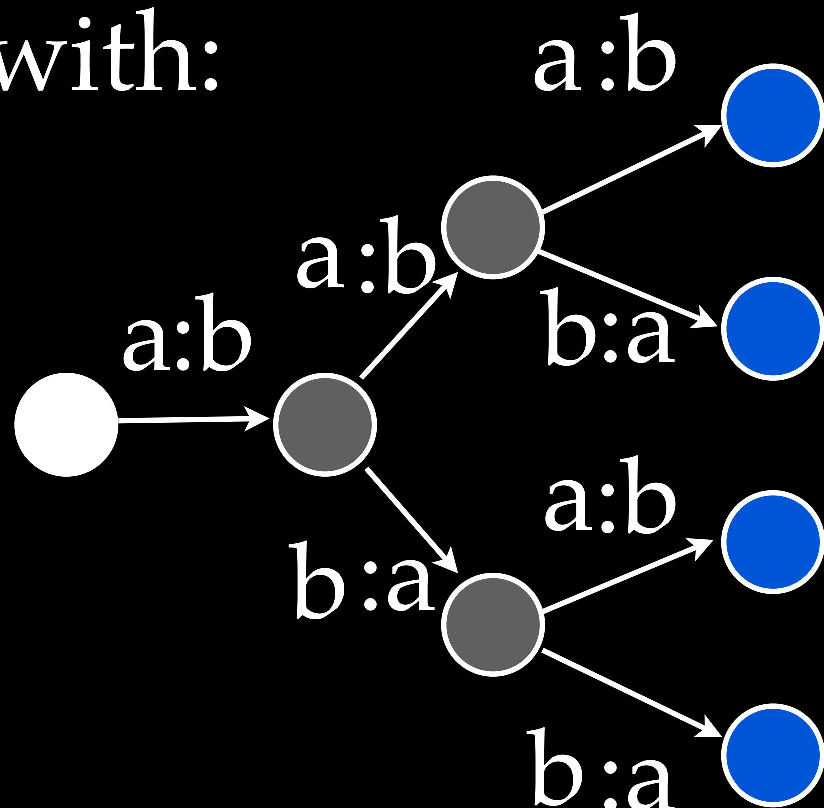
Composing Transducers



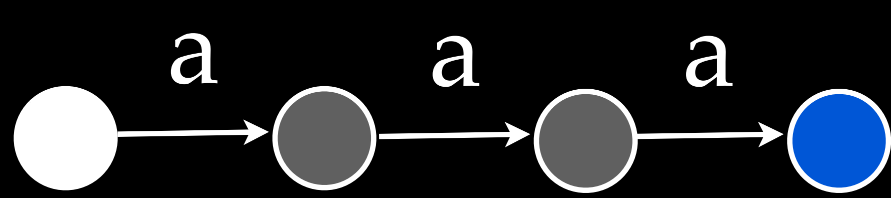
Composing Transducers



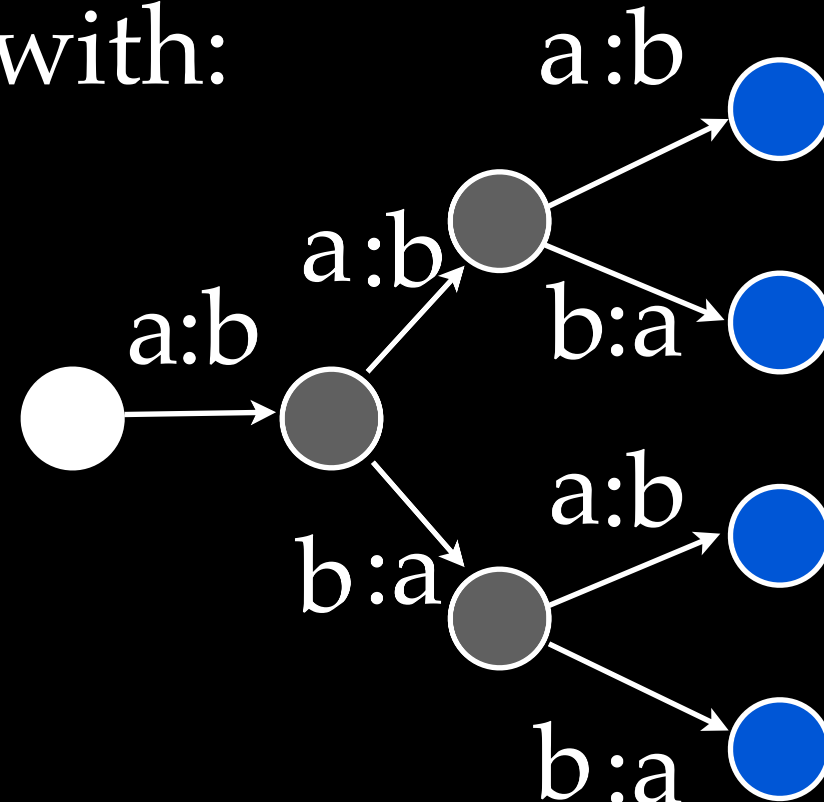
composed with:



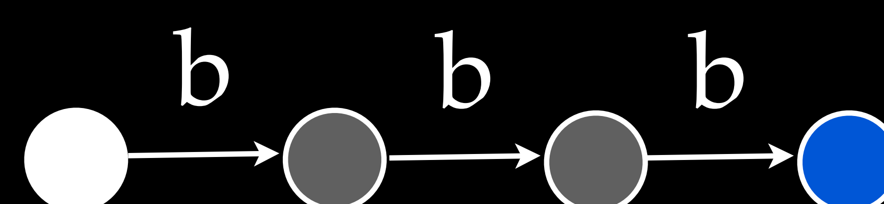
Composing Transducers



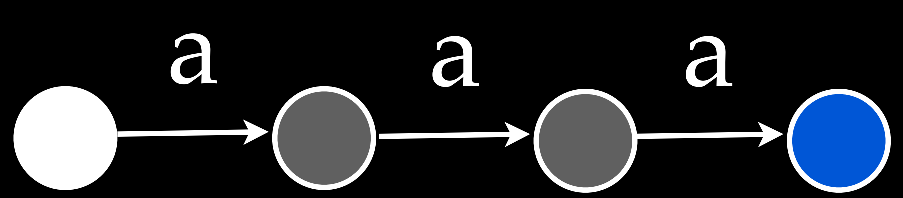
composed with:



yields:



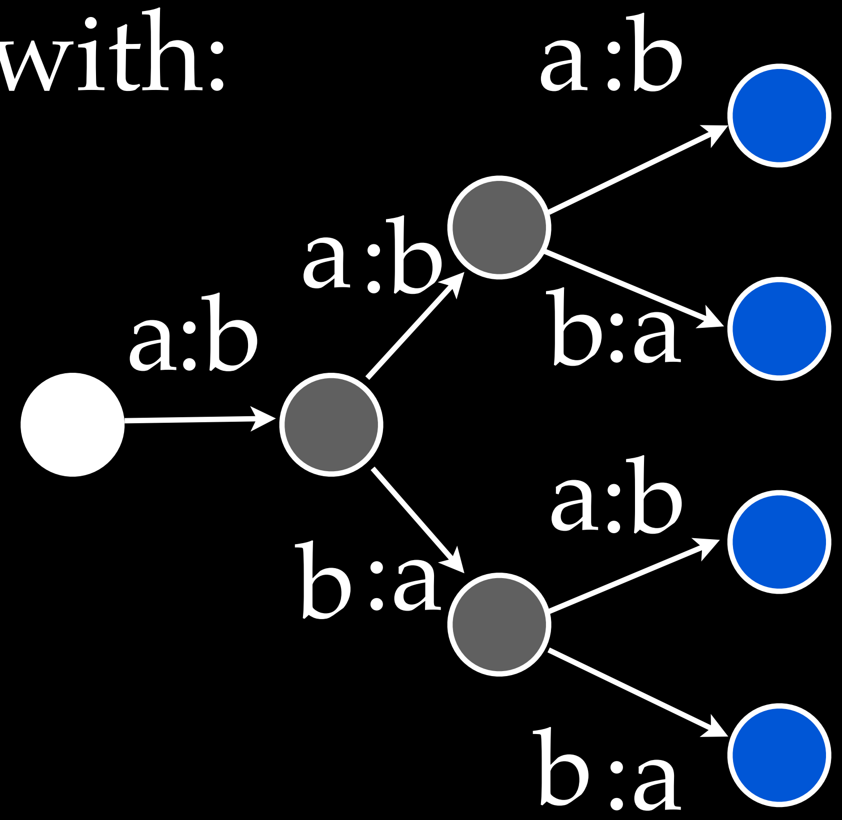
Composing Transducers



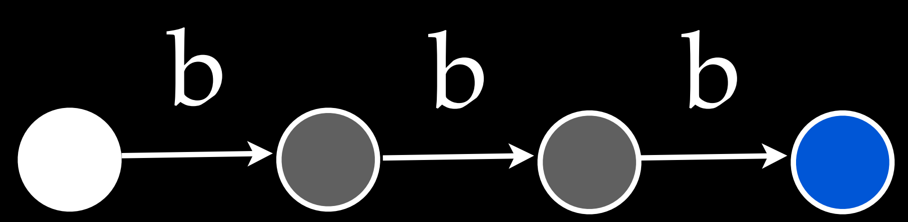
computes a function:

$$f : \mathcal{L}_1 \rightarrow \mathcal{P}(\mathcal{L}_2)$$

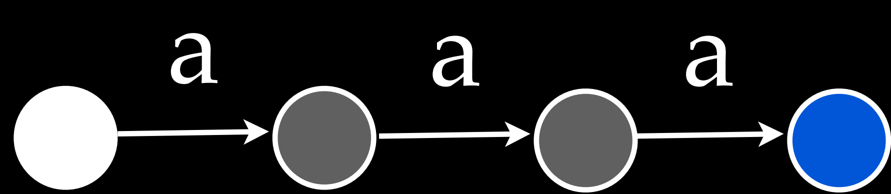
composed with:



yields:



Composing Transducers



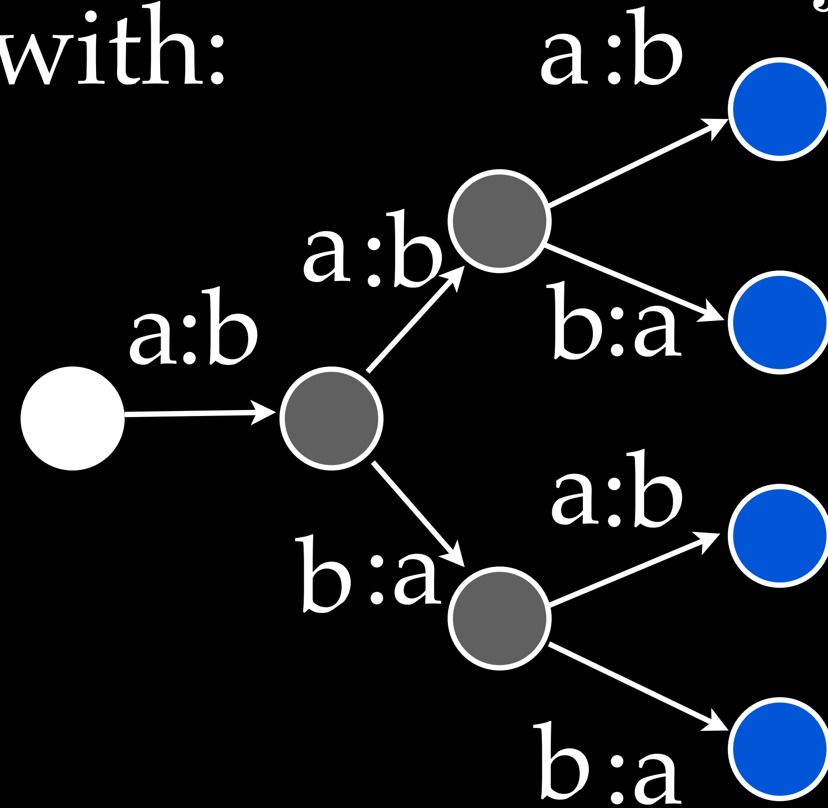
computes a function:

$$f : \mathcal{L}_1 \rightarrow \mathcal{P}(\mathcal{L}_2)$$

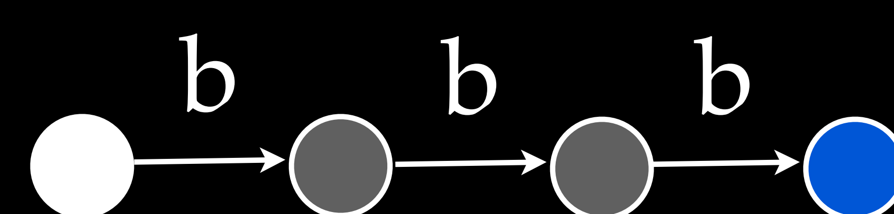
with weights:

$$f : \mathcal{L}_1 \rightarrow \mathcal{P}(\mathcal{L}_2 \times \mathbb{R}^+)$$

composed with:



yields:



IBM Model 4

IBM Model 4

Although north wind howls , but sky still very clear .

虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。

IBM Model 4

Although north wind howls , but sky still very clear .

虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。

↓

虽然

IBM Model 4

Although north wind howls , but sky still very clear .

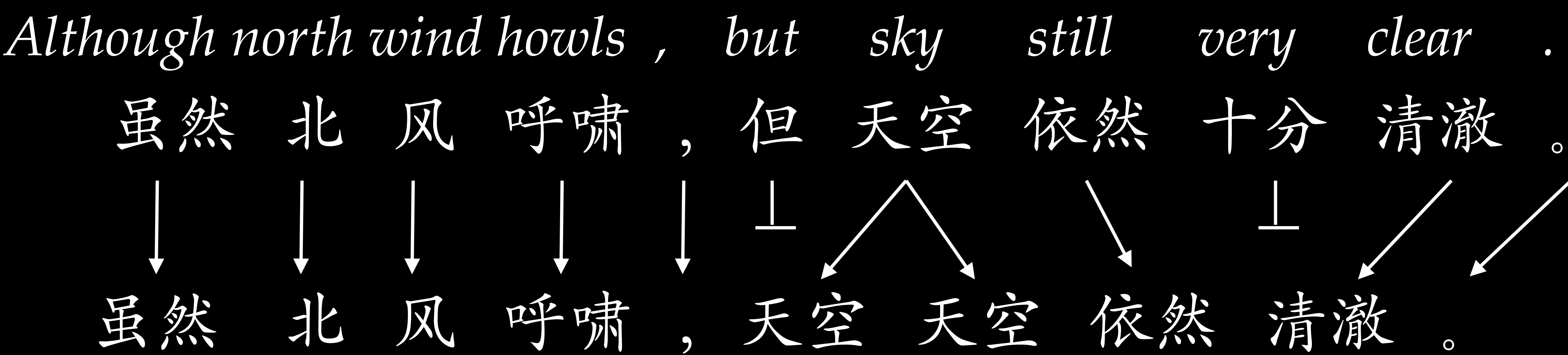
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。

↓

虽然

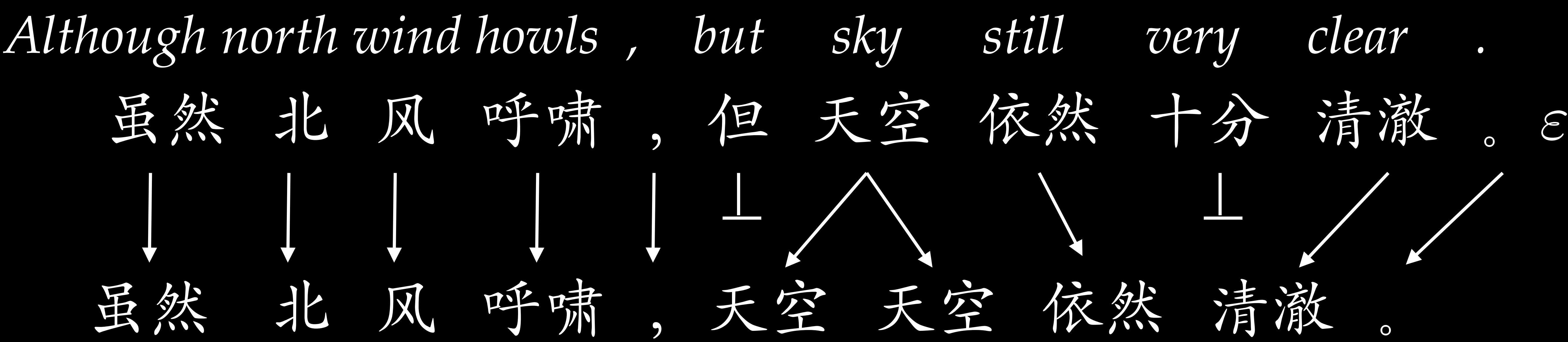
$p_f(1|\text{虽然})$

IBM Model 4



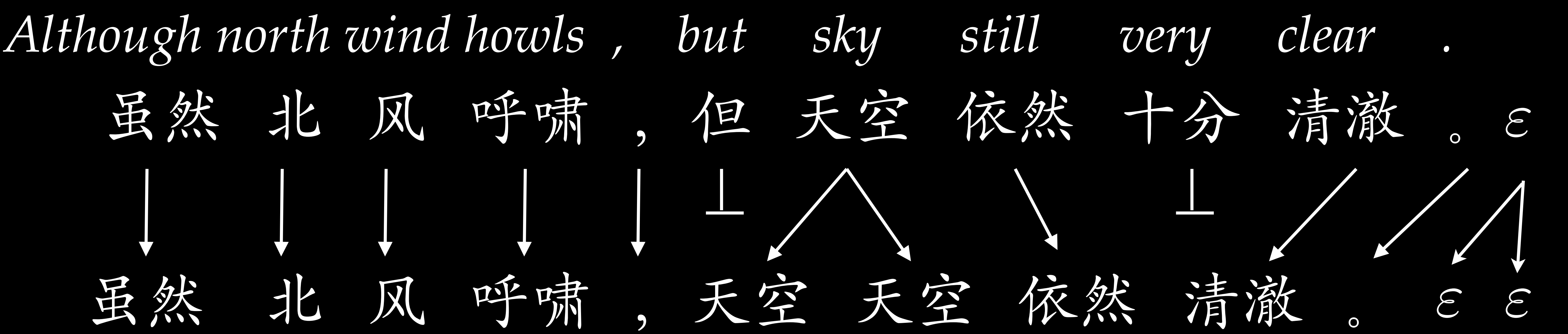
Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。
⊥
⊥
虽然 北 风 呼啸 ，天空 天空 依然 清澈 。

IBM Model 4

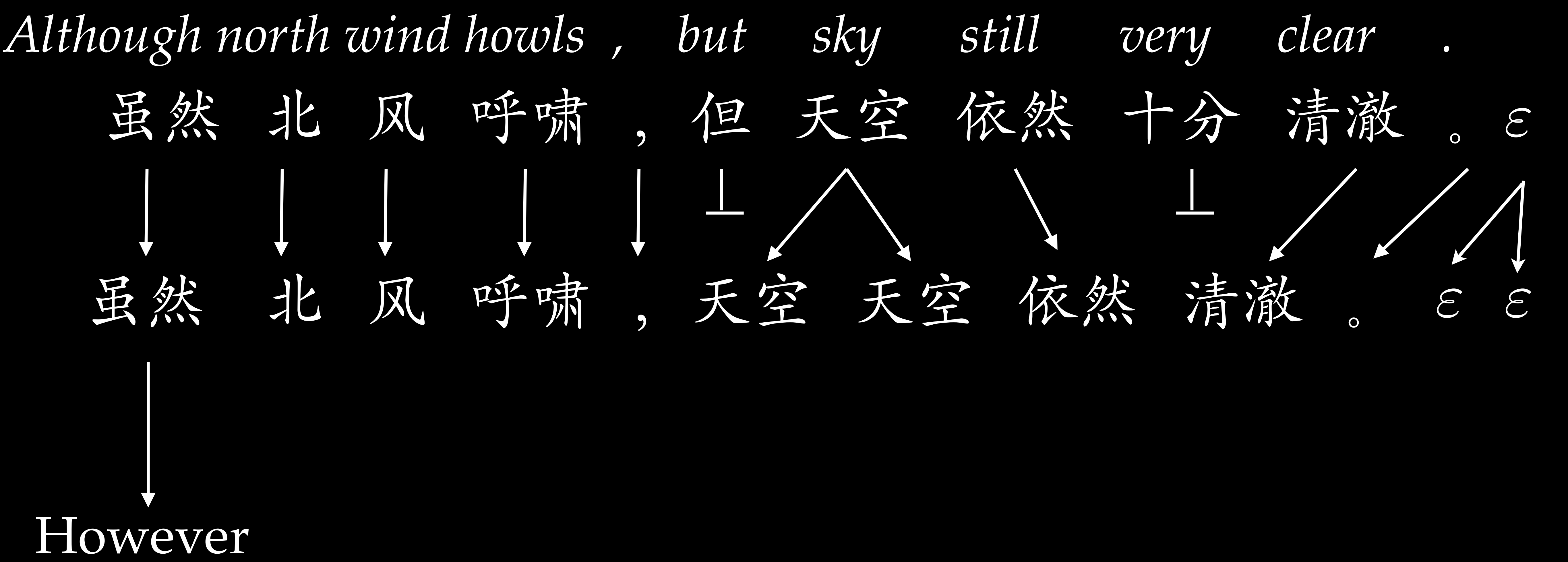


Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。ε
⊥
⊥
虽然 北 风 呼啸 ，天空 天空 依然 清澈 。

IBM Model 4


Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。ε
⊥
⊥
虽然 北 风 呼啸 ，天空 天空 依然 清澈 。 ε ε

IBM Model 4

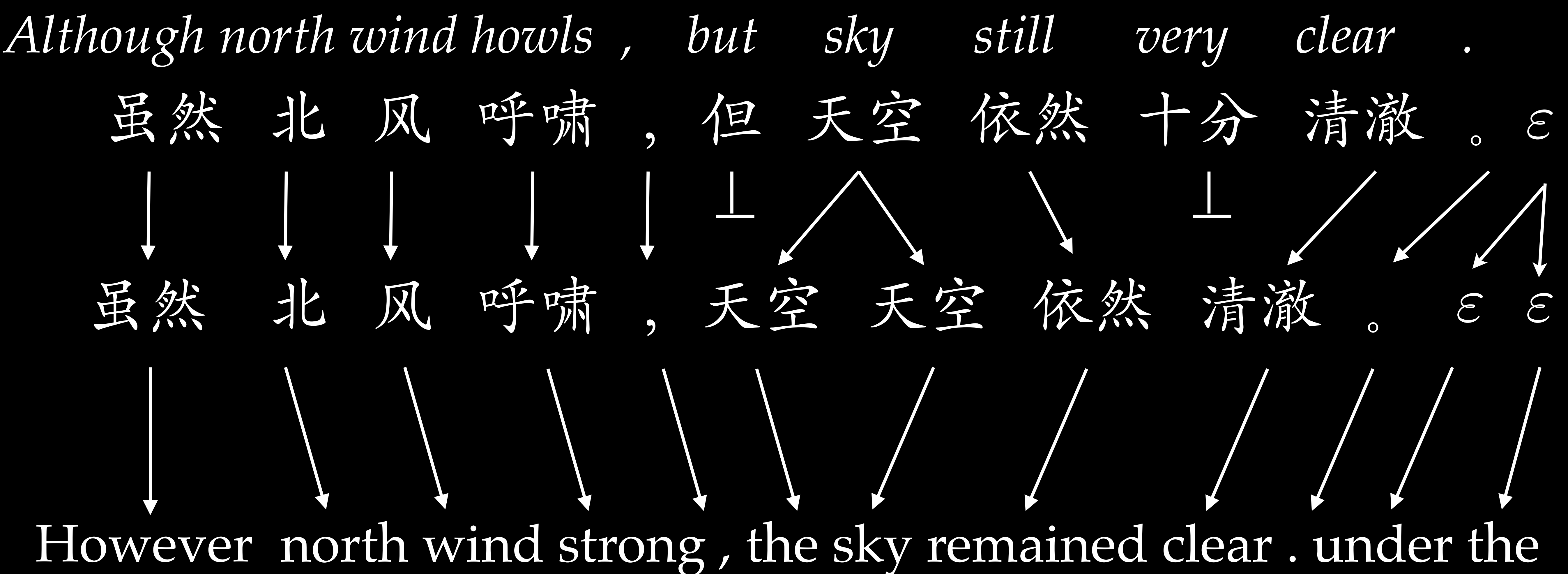


Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ， 但 天空 依然 十分 清澈 。 ε
⊥ ⊥
虽然 北 风 呼啸 ， 天空 天空 依然 清澈 。 ε ε
However

IBM Model 4

$$p_t(However|虽然)$$

IBM Model 4



Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。ε
⊥ ⊥
虽然 北 风 呼啸 ，天空 天空 依然 清澈 。 ε ε
However north wind strong , the sky remained clear . under the

IBM Model 4


Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ， 但 天空 依然 十分 清澈 。 ε
⊥ ⊥
虽然 北 风 呼啸 ， 天空 天空 依然 清澈 。 ε ε
However north wind strong , the sky remained clear . under the

IBM Model 4

Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ， 但 天空 依然 十分 清澈 。 ε
⊥
⊥
虽然 北 风 呼啸 ， 天空 天空 依然 清澈 。 ε ε
However north wind strong , the sky remained clear . under the

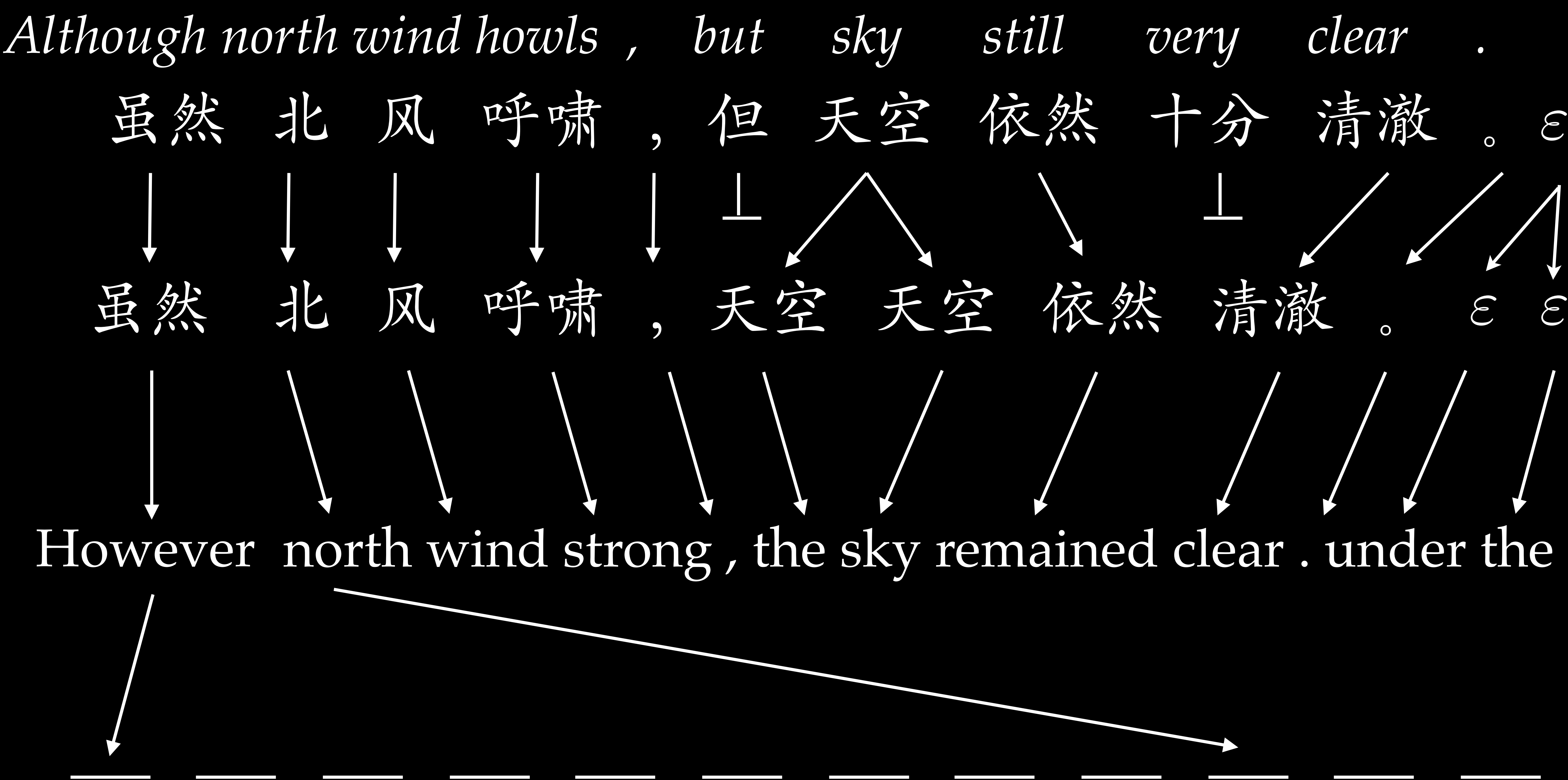
IBM Model 4

$$p_d(0|However)$$

IBM Model 4

Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ， 但 天空 依然 十分 清澈 。 ε
⊥ ⊥
虽然 北 风 呼啸 ， 天空 天空 依然 清澈 。 ε ε
However north wind strong , the sky remained clear . under the

IBM Model 4



$$p_d(8|north)$$

IBM Model 4


Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。ε
⊥
⊥
虽然 北 风 呼啸 ，天空 天空 依然 清澈 。 ε ε
However north wind strong , the sky remained clear . under the

IBM Model 4

Although north wind howls , but sky still very clear .
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。ε
⊥
⊥
虽然 北 风 呼啸 ，天空 天空 依然 清澈 。 ε ε
However north wind strong , the sky remained clear . under the
However , the sky remained clear under the strong north wind .

IBM Model 4

However , the sky remained clear under the strong north wind .

$$p(English, alignment|Chinese) = \prod_{p_f} \prod_{p_t} \prod_{p_d}$$

IBM Model 4

Although north wind howls , but sky still very clear .

虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。

IBM Model 4

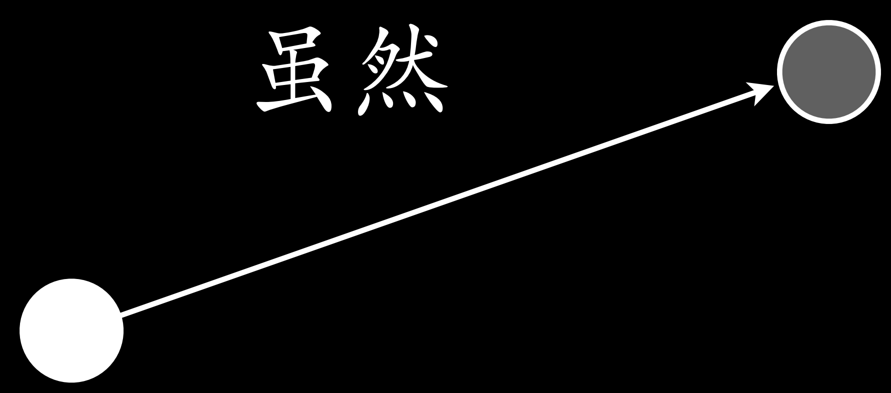
Although north wind howls , but sky still very clear .

虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。

IBM Model 4

Although north wind howls , but sky still very clear .

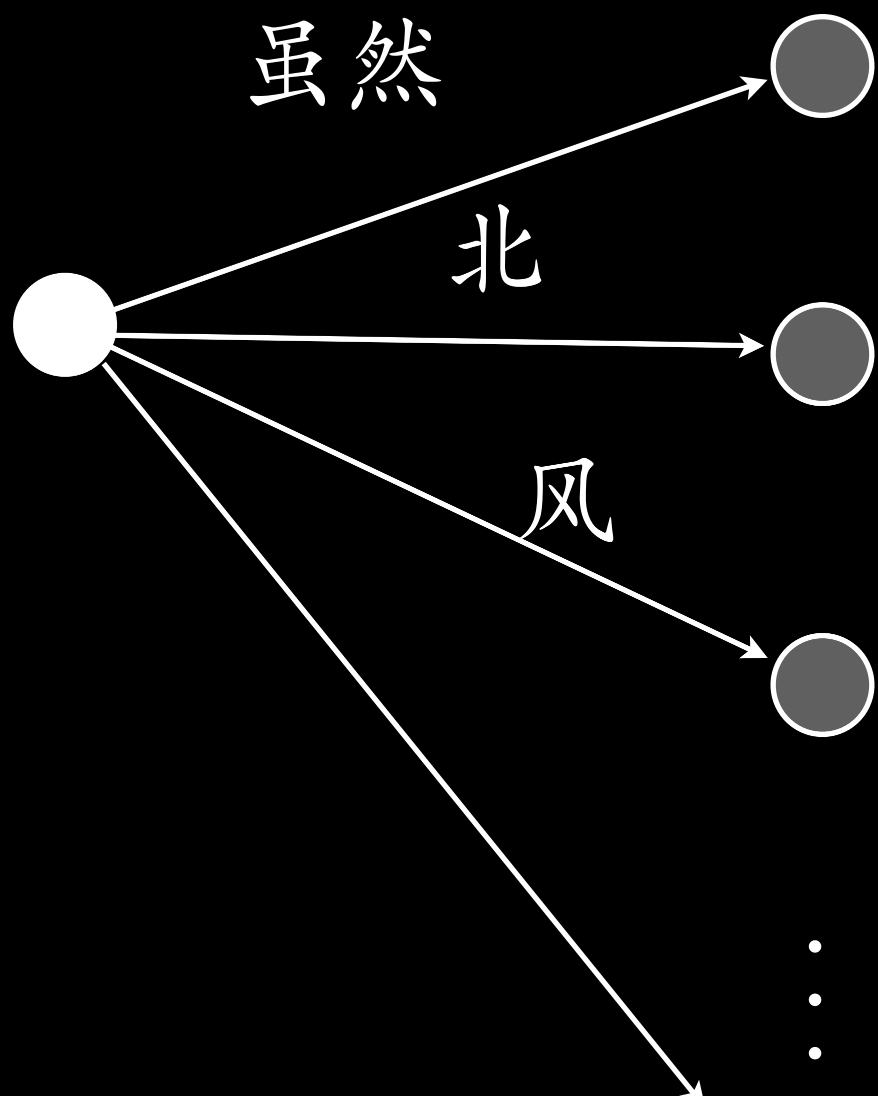
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。



IBM Model 4

Although north wind howls , but sky still very clear .

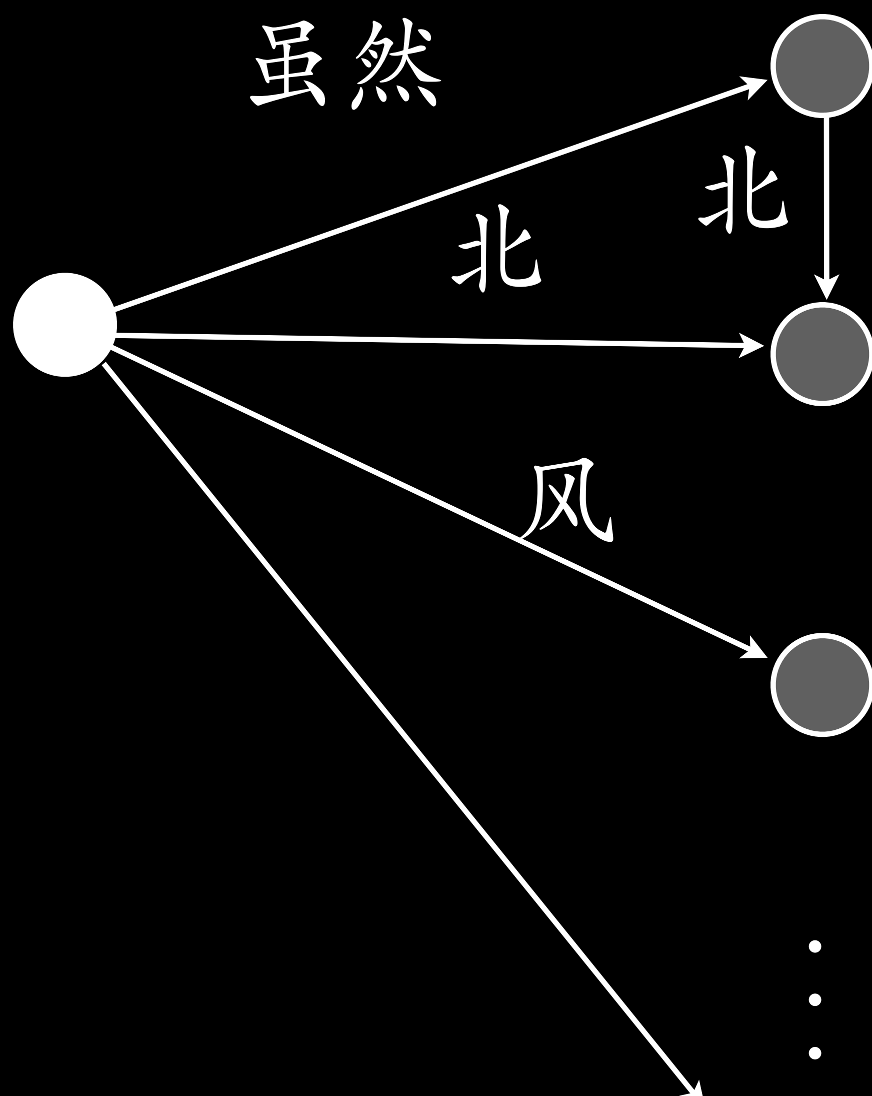
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。



IBM Model 4

Although north wind howls , but sky still very clear .

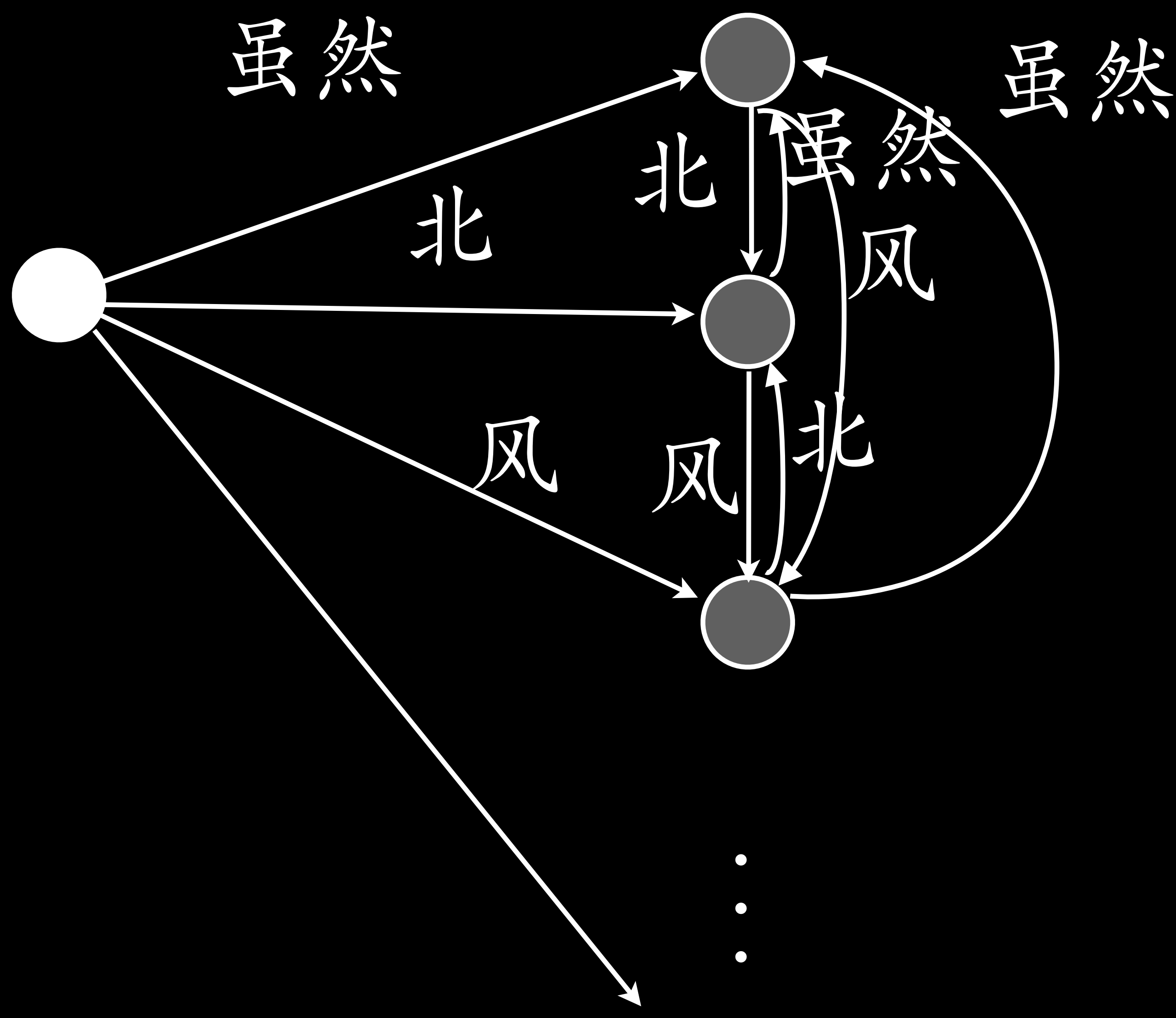
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。



IBM Model 4

Although north wind howls , but sky still very clear .

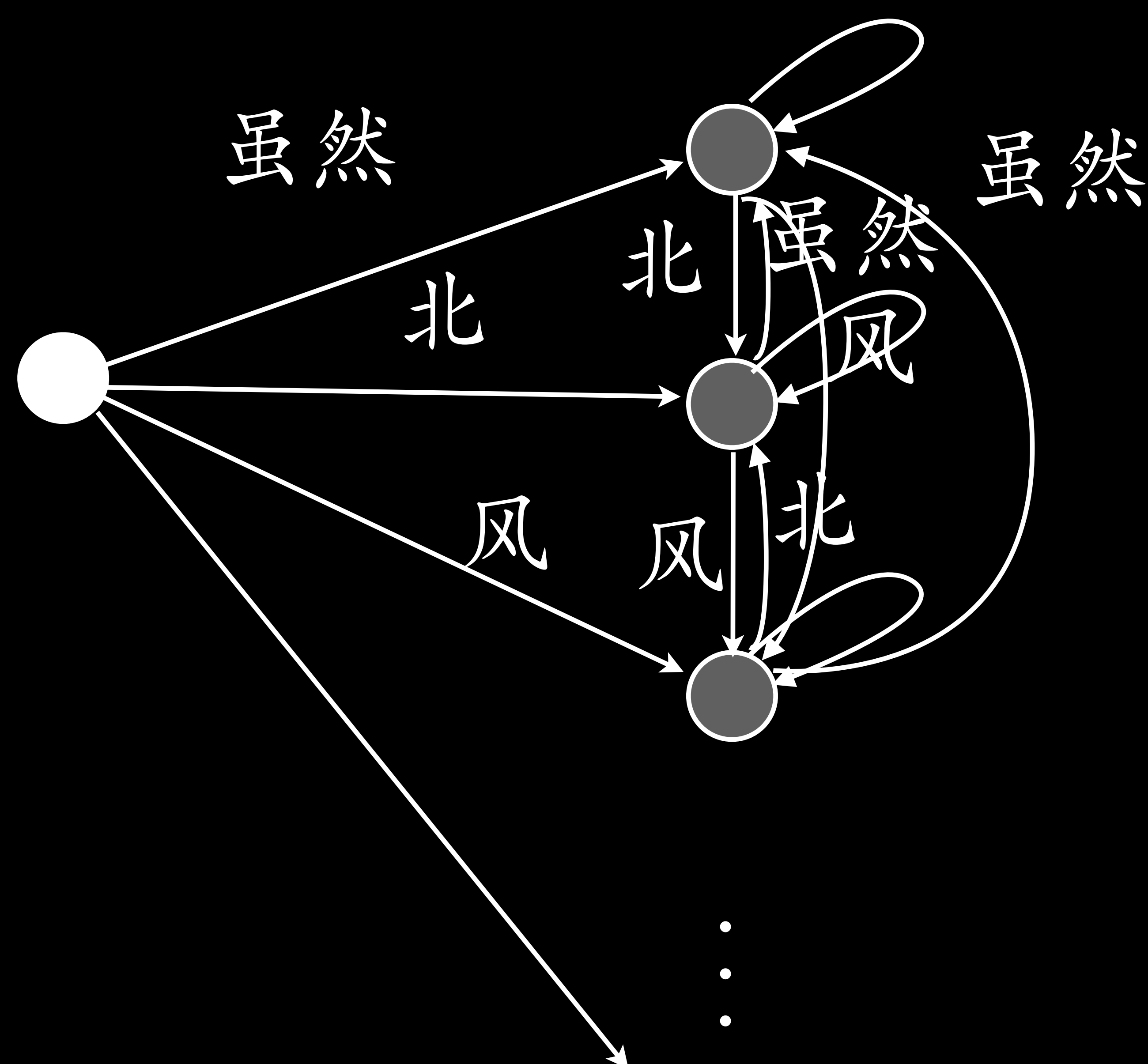
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。



IBM Model 4

Although north wind howls , but sky still very clear .

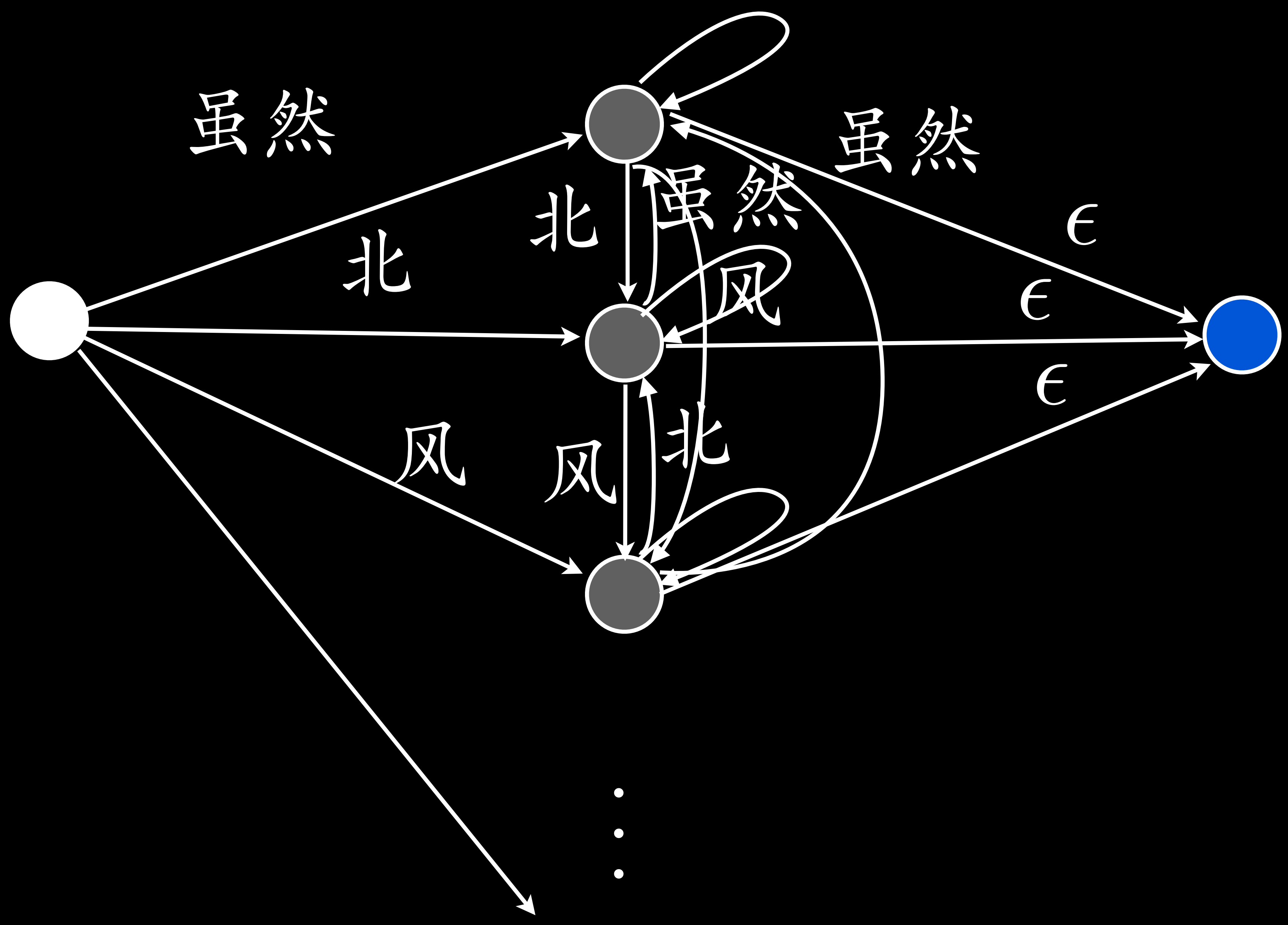
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。



IBM Model 4

Although north wind howls , but sky still very clear .

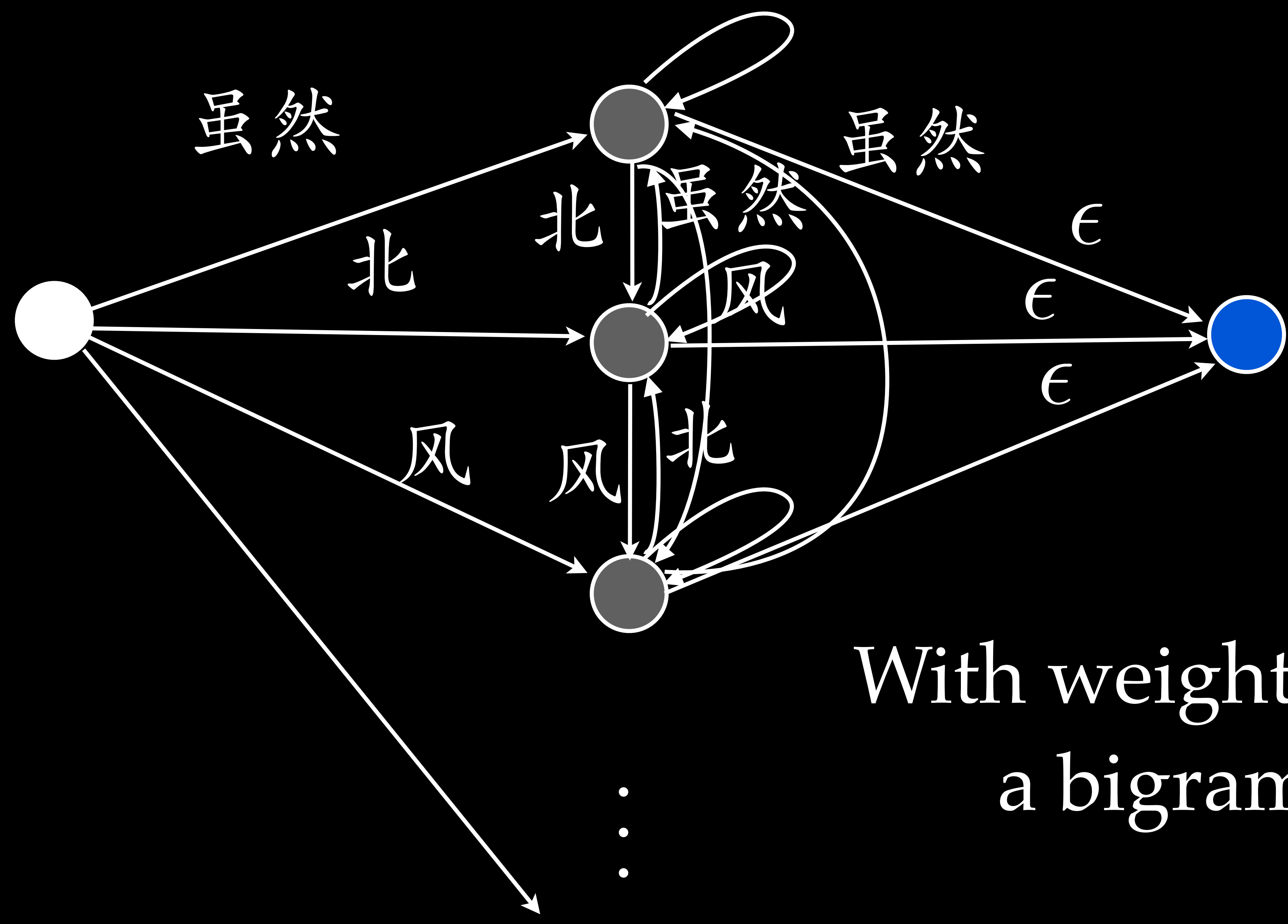
虽然 北 风 呼啸 ， 但 天空 依然 十分 清澈 。



IBM Model 4

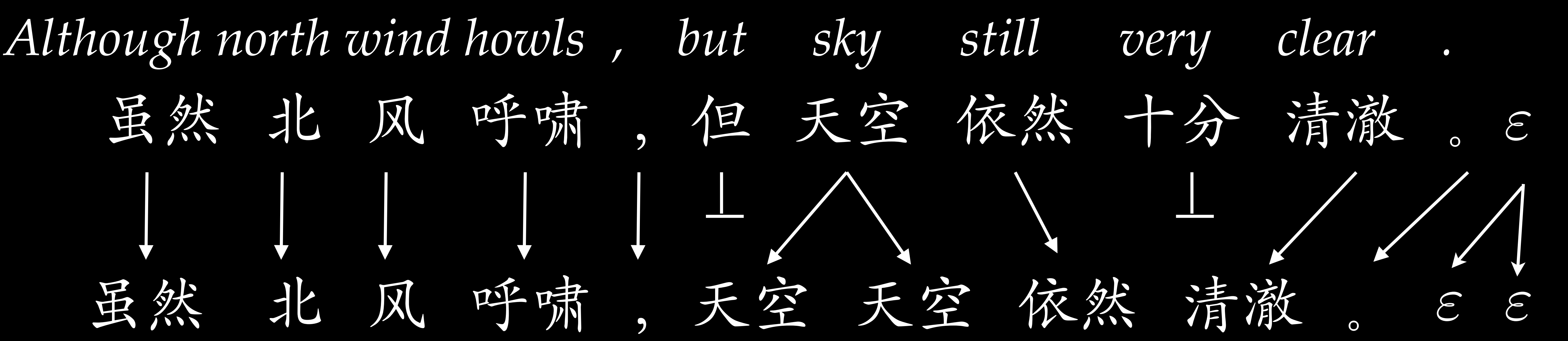
Although north wind howls , but sky still very clear .

虽然 北 风 呼啸 ， 但 天空 依然 十分 清澈 。

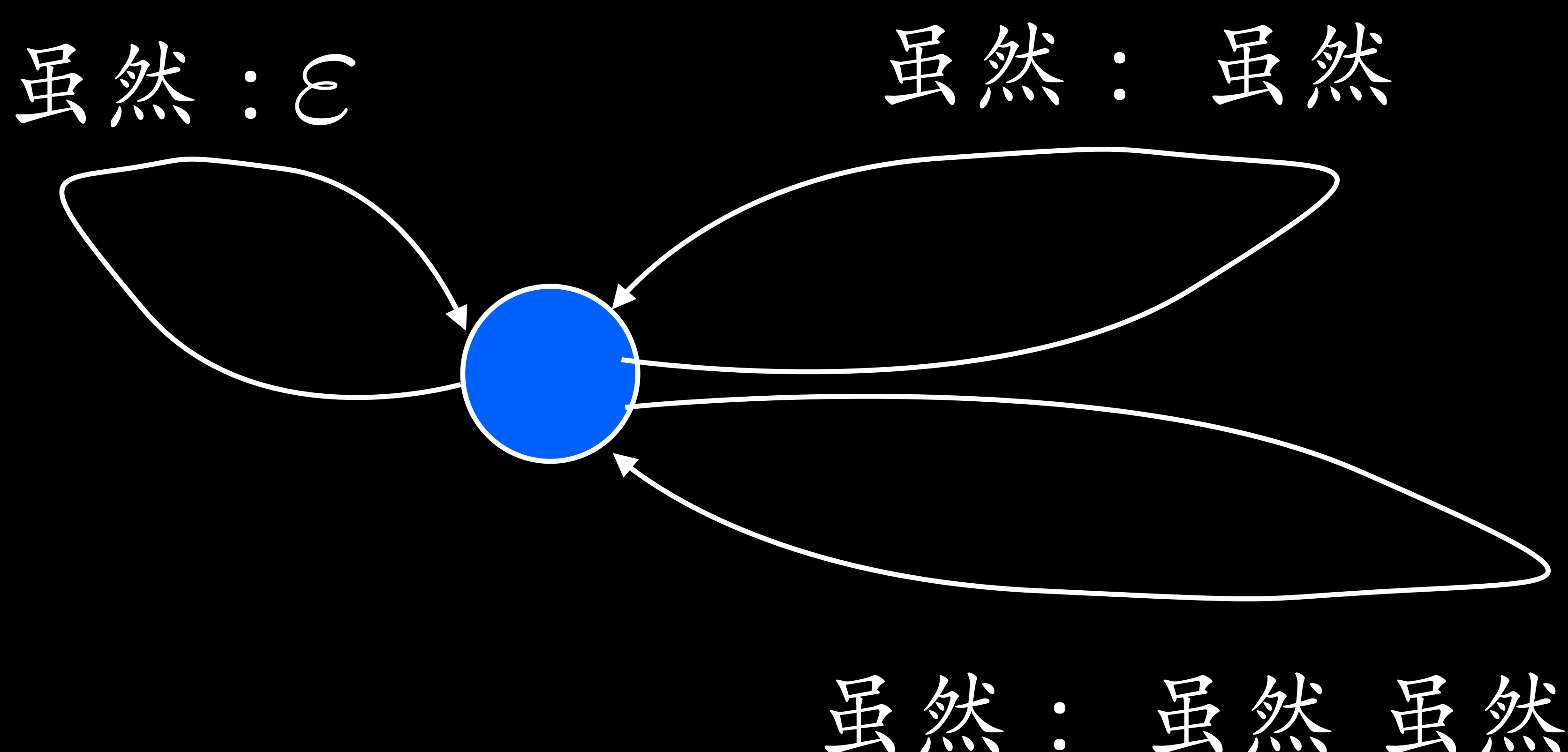


With weights, defines a bigram LM!

IBM Model 4



fertility transducer

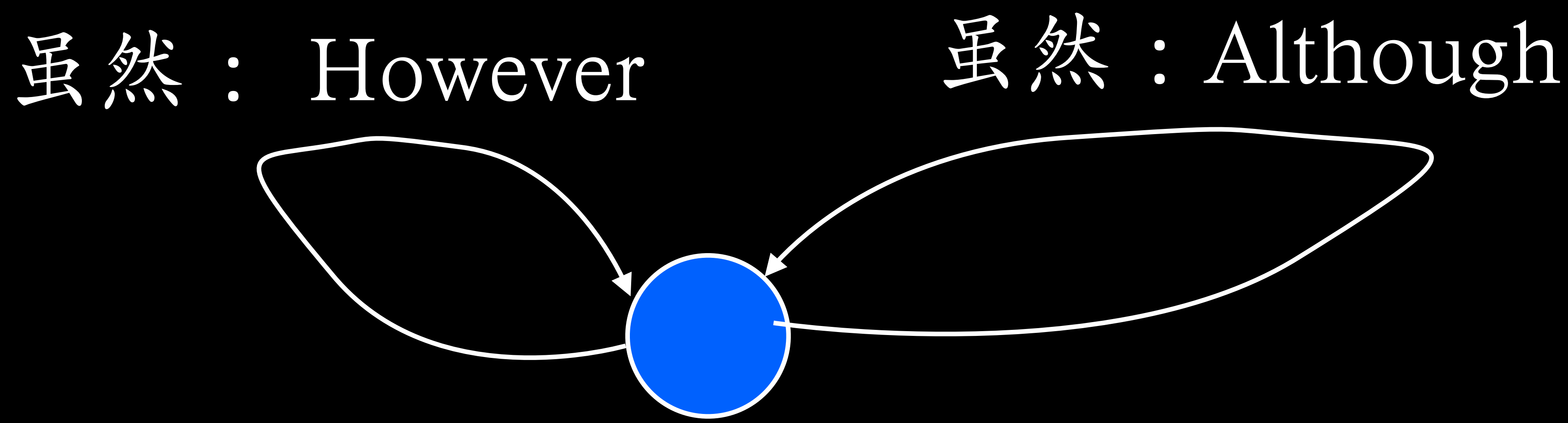


IBM Model 4

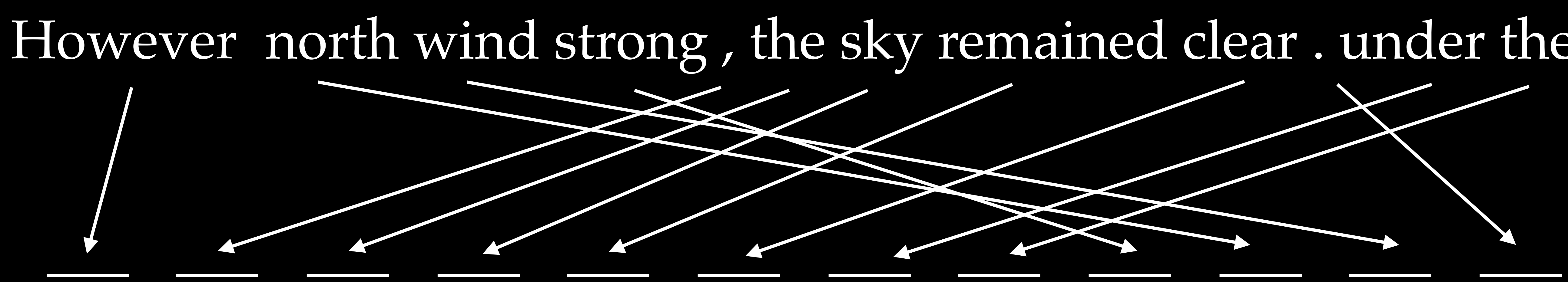

虽然 北 风 呼啸 ， 天空 天空 依然 清澈 。 ε ε
However north wind strong , the sky remained clear . under the

IBM Model 4

substitution transducer

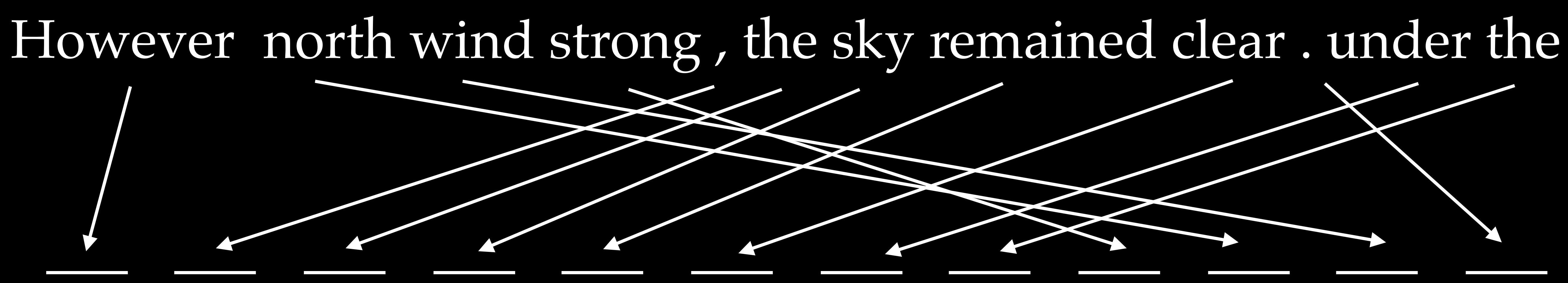


IBM Model 4



IBM Model 4

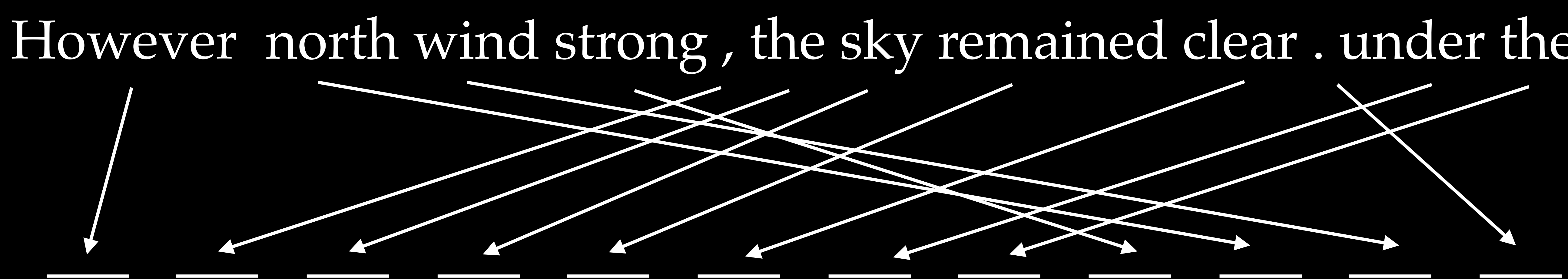
This is not a transducer, but we can build an FSA that accepts all permutations.



IBM Model 4

This is not a transducer, but we can build an FSA that accepts all permutations.

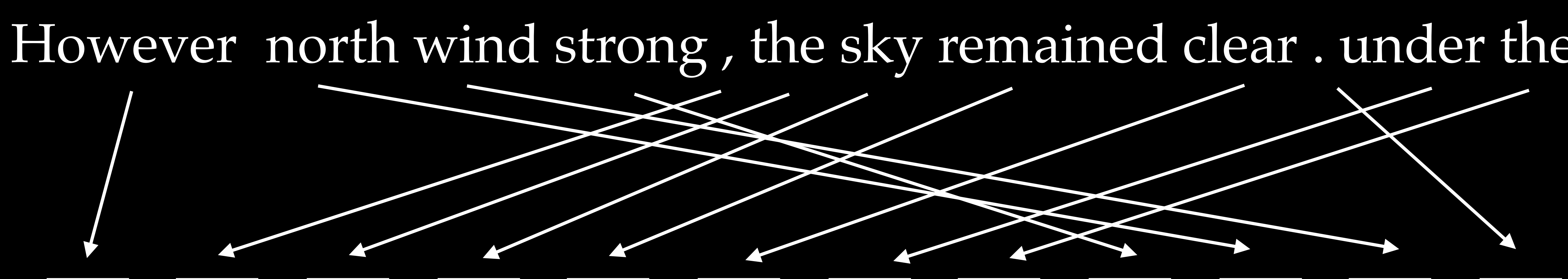
To translate: run the entire cascade in reverse!



IBM Model 4

This is not a transducer, but we can build an FSA that accepts all permutations.

To translate: run the entire cascade in reverse!



Looks like stack decoding algorithm Matt showed you, but doesn't know as much about graph topology.

Phrase-based Models

Although north wind howls , but sky still very clear .

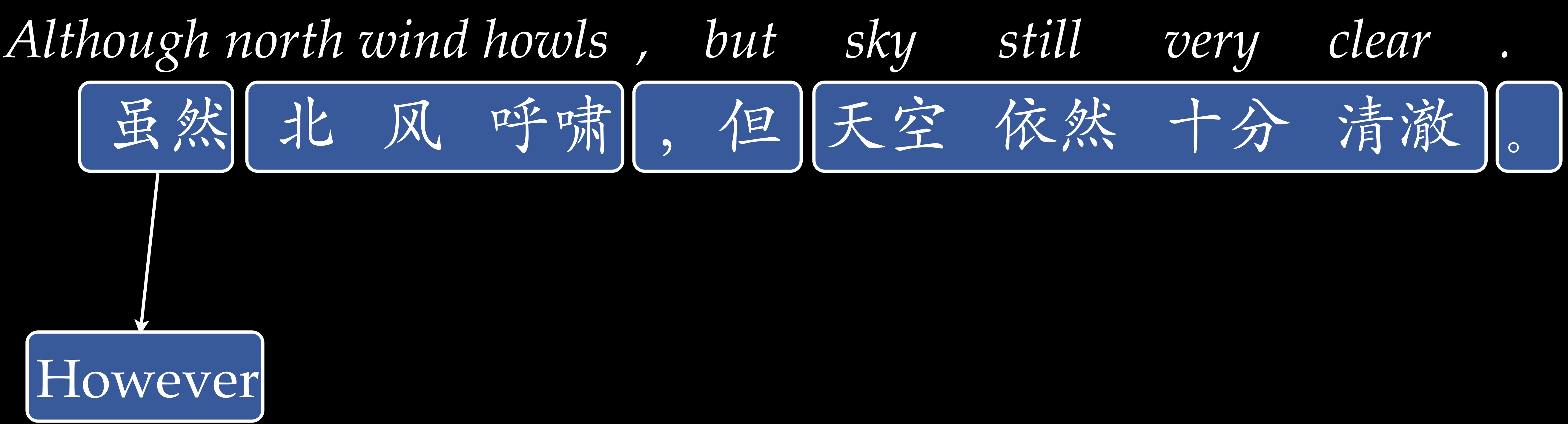
虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。

Phrase-based Models

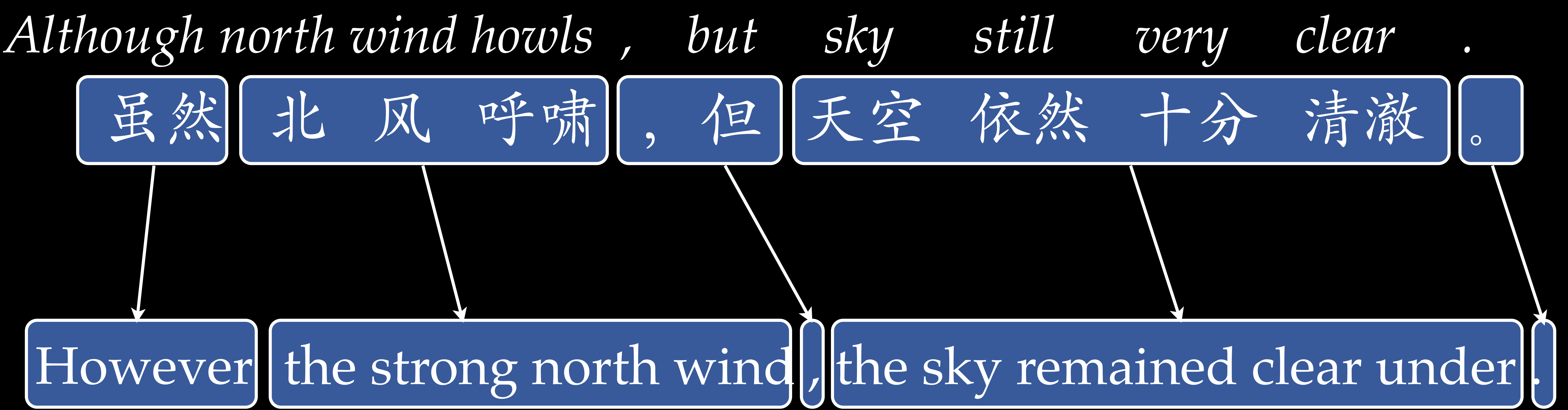
Although north wind howls , but sky still very clear .

虽然	北 风 呼啸	，但	天空 依然 十分 清澈	。

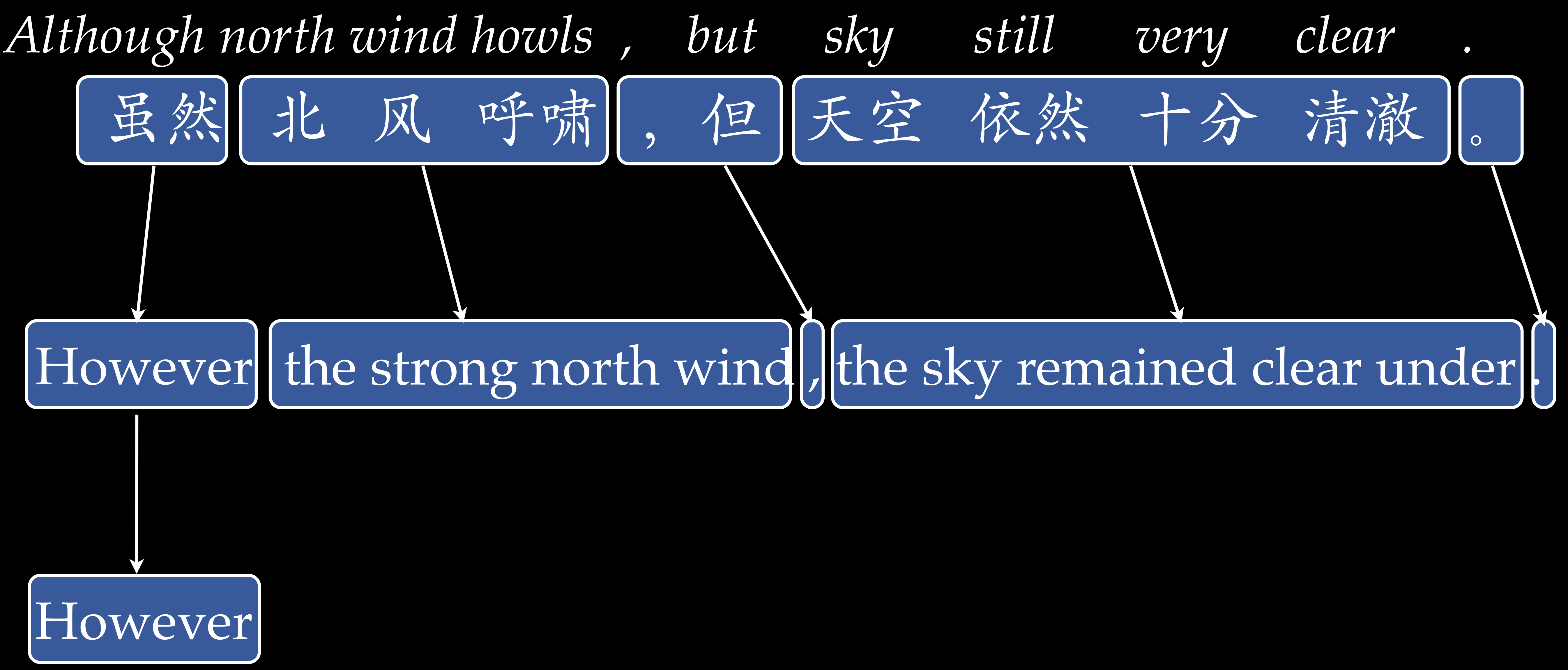
Phrase-based Models



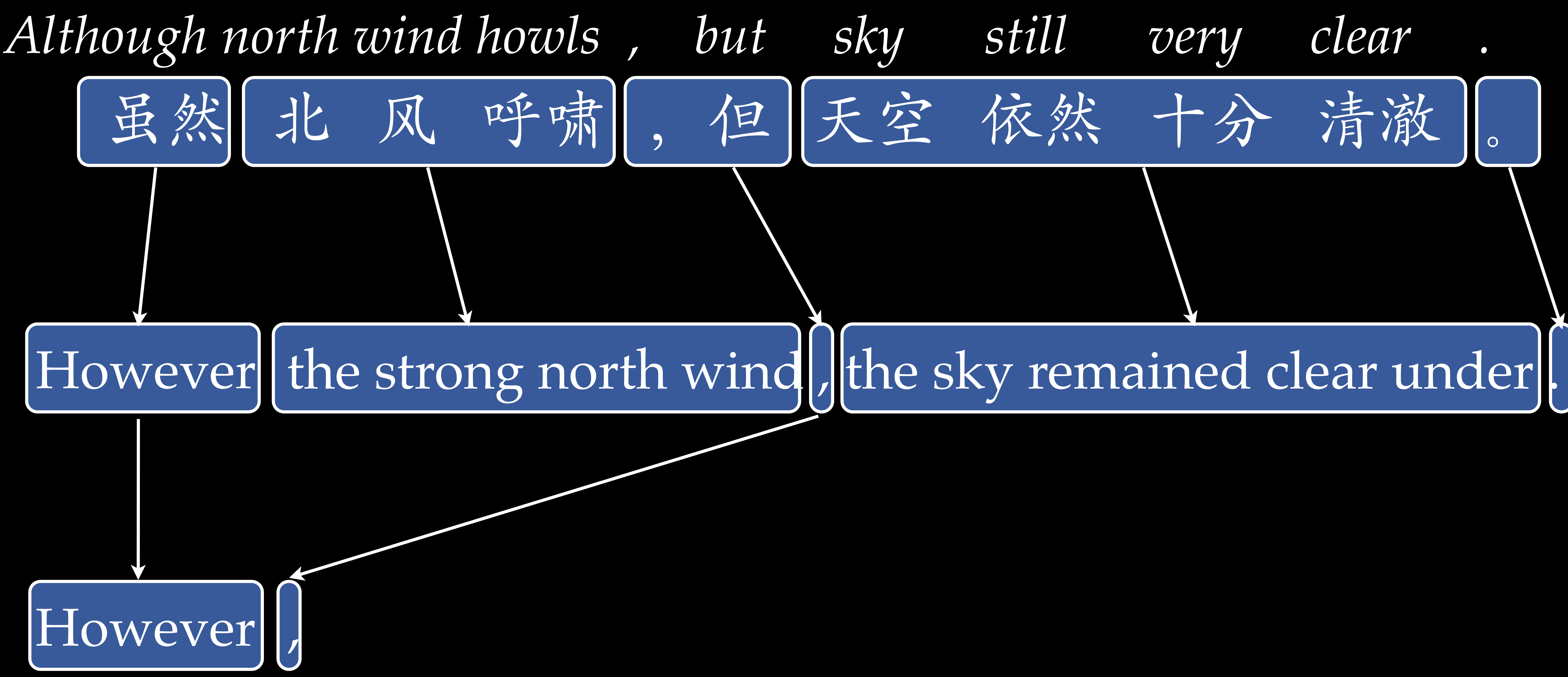
Phrase-based Models



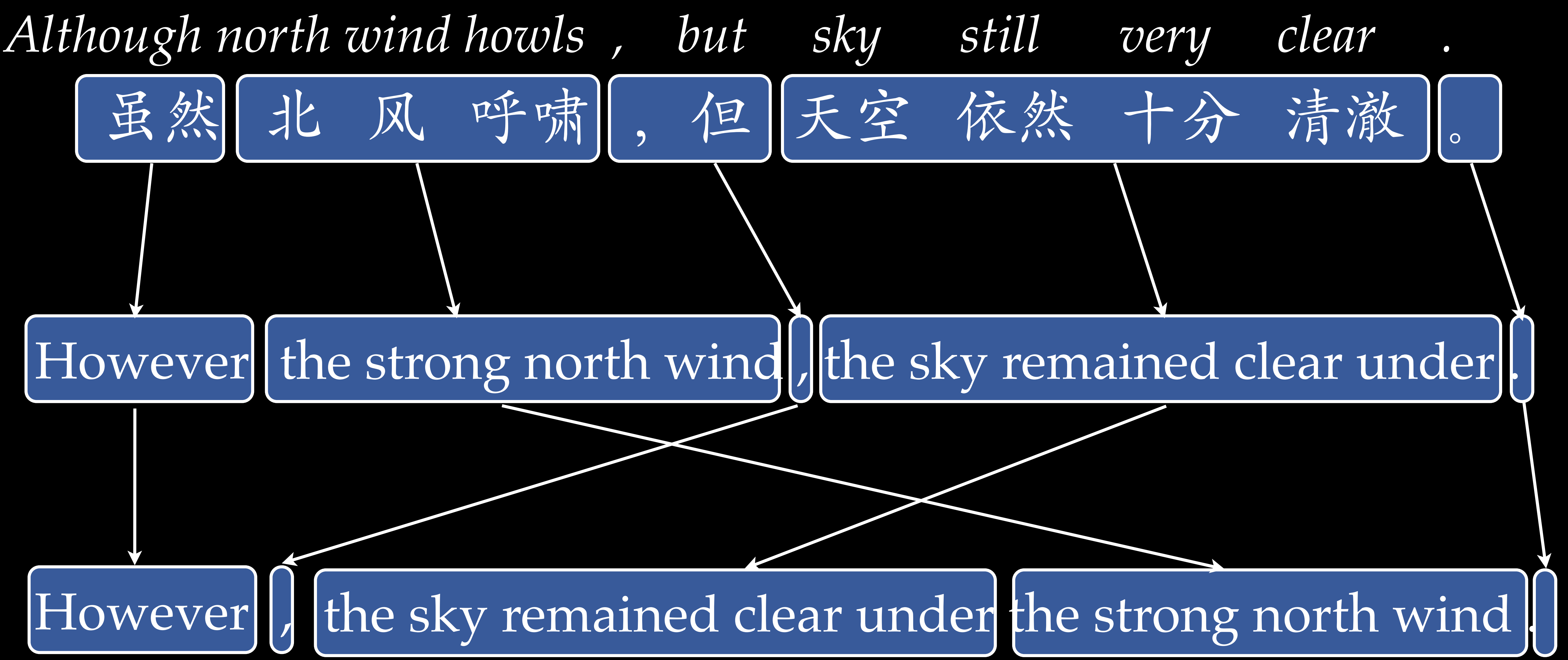
Phrase-based Models



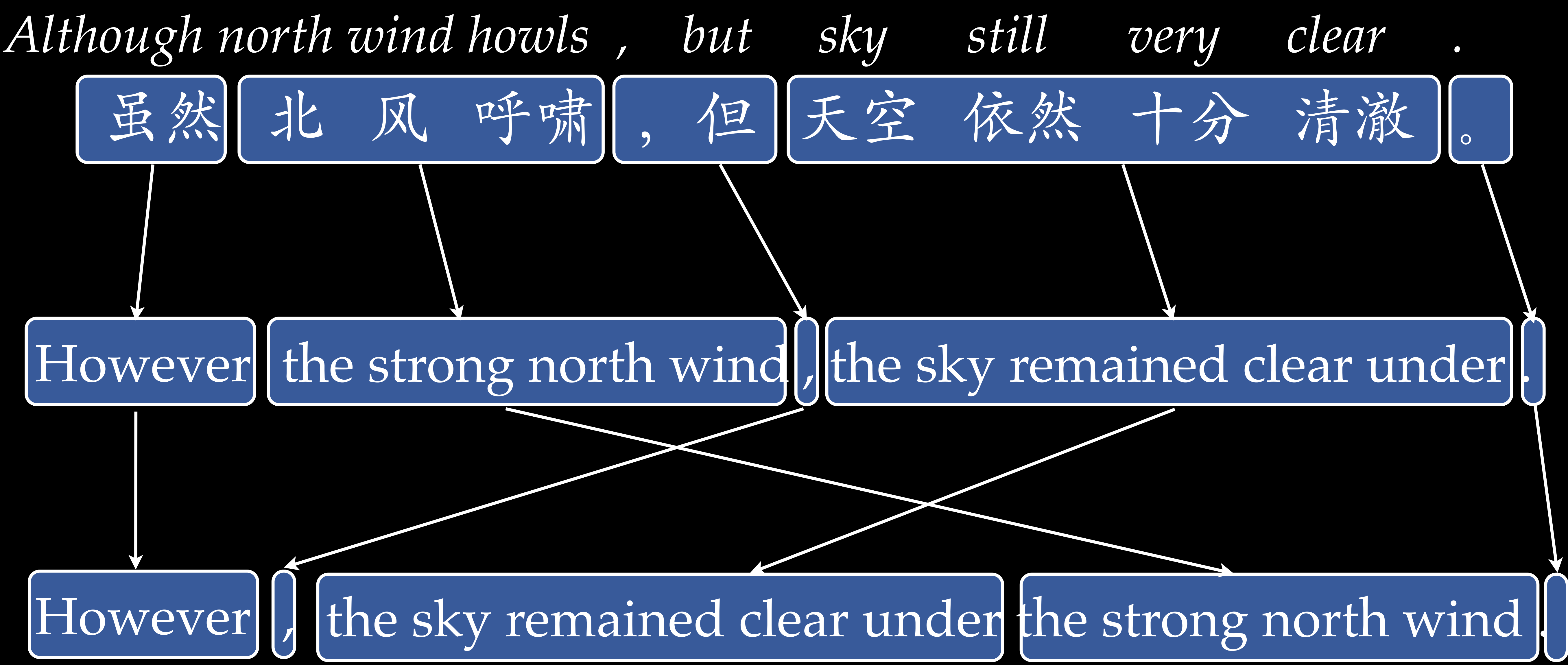
Phrase-based Models



Phrase-based Models



Phrase-based Models



$$p(English, alignment|Chinese) =$$

$$p(segmentation) \cdot p(translations) \cdot p(reorderings)$$

Phrase-based Models

Although north wind howls , but sky still very clear .

虽然 北 风 呼啸 ，但 天空 依然 十分 清澈 。

Phrase-based Models

Although north wind howls , but sky still very clear .

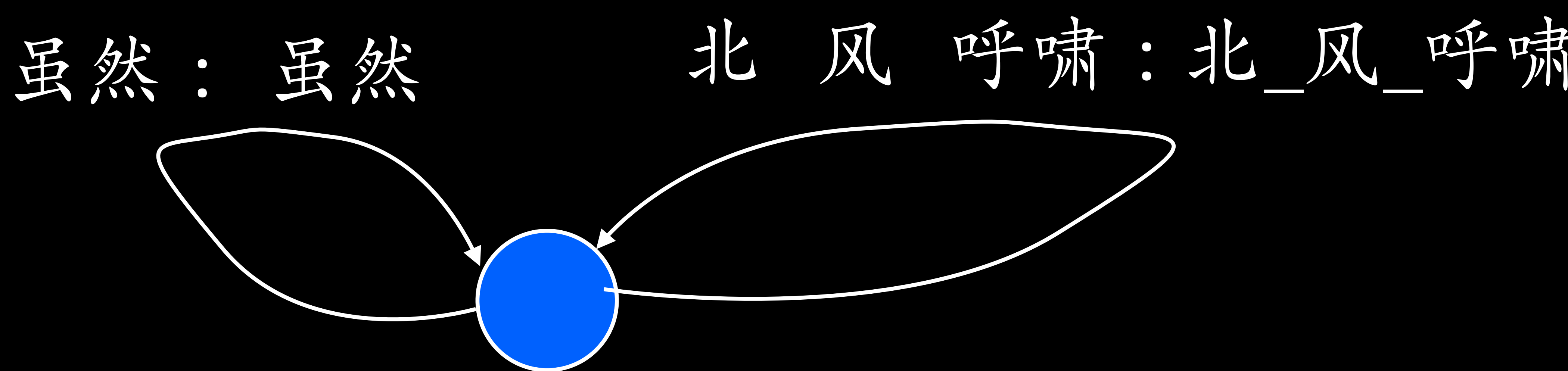
虽然	北 风 呼啸	，但	天空 依然 十分 清澈	。

Phrase-based Models

Although north wind howls , but sky still very clear .

虽然	北 风 呼啸	，但	天空 依然 十分 清澈	。

segmentation transducer

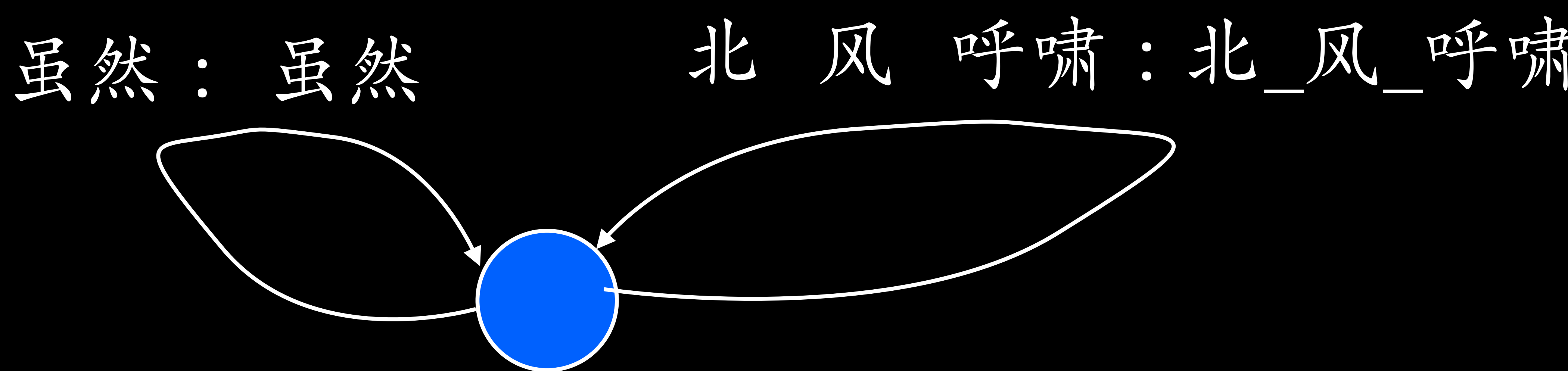


Phrase-based Models

Although north wind howls , but sky still very clear .

虽然 | 北 风 呼啸 | ， 但 | 天空 依然 十分 清澈 | 。

segmentation transducer



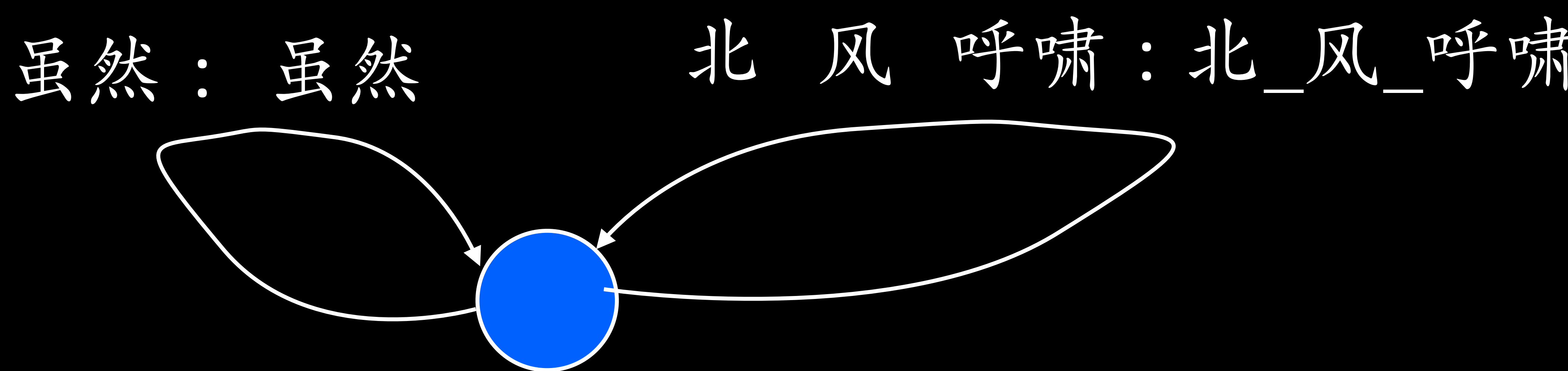
As with IBM Models, construct a cascade of transducers.

Phrase-based Models

Although north wind howls , but sky still very clear .

虽然 | 北 风 呼啸 | ，但 | 天空 依然 十分 清澈 | 。

segmentation transducer



As with IBM Models, construct a cascade of transducers.

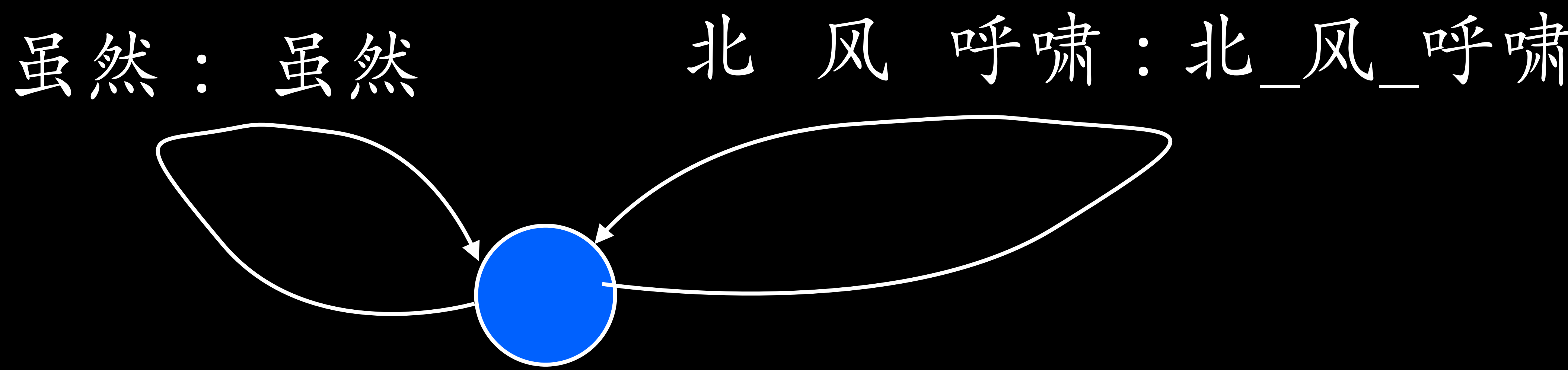
Determinize, remove epsilons, minimize at each step.

Phrase-based Models

Although north wind howls , but sky still very clear .

虽然 | 北 风 呼啸 | ，但 | 天空 依然 十分 清澈 | 。

segmentation transducer

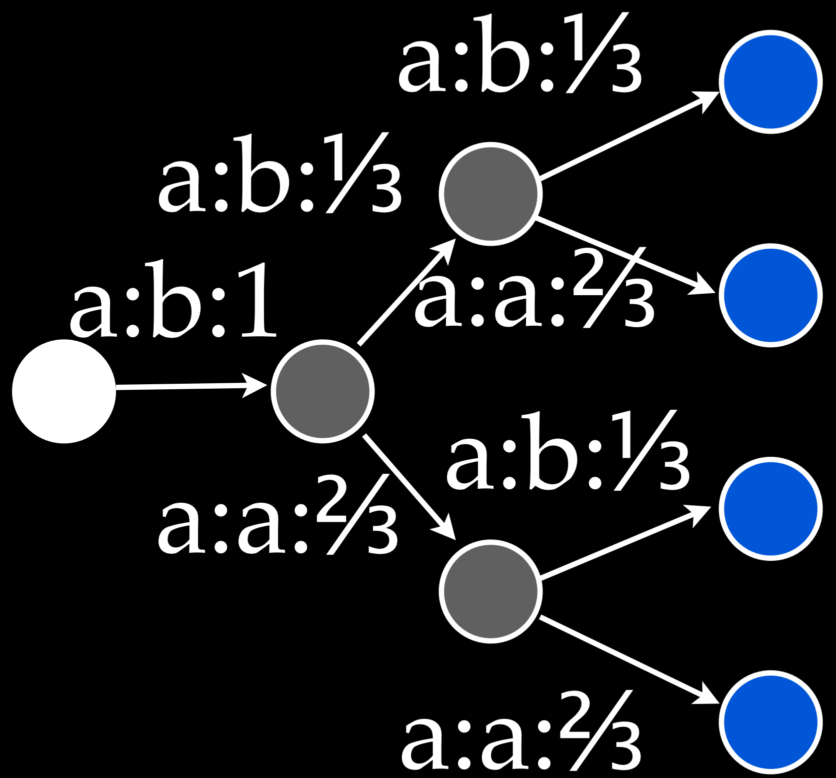


As with IBM Models, construct a cascade of transducers.

Determinize, remove epsilons, minimize at each step.

See Kumar et al. (2004) for a complete implementation.

A Note on *Weights*



$\langle$ a a a, b b b $\rangle \to$ 1/9

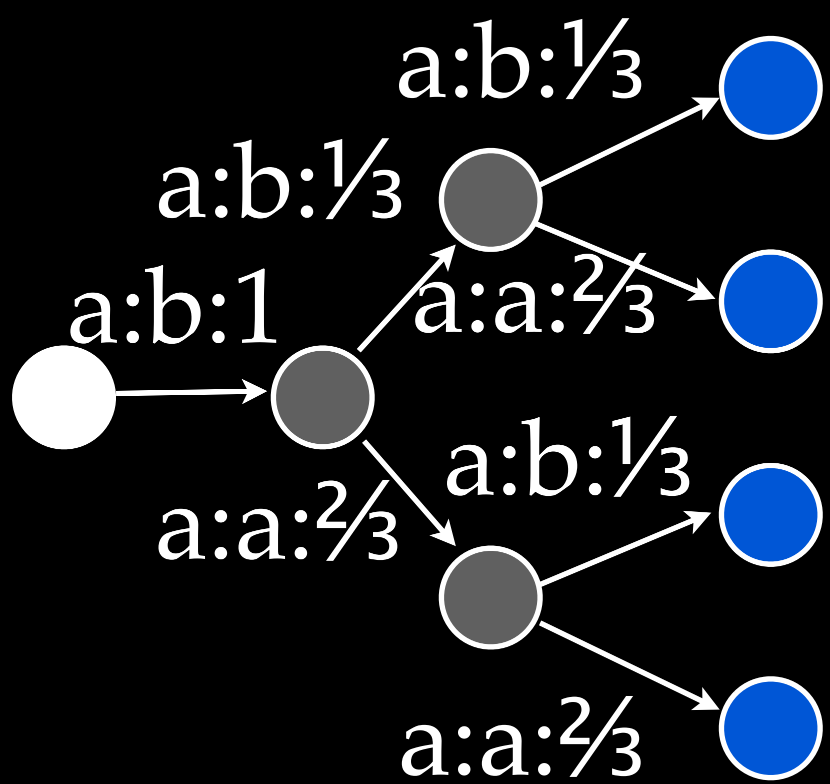
$\langle$ a a a, b a b $\rangle \to$ 2/9

$\langle$ a a a, b b a $\rangle \to$ 2/9

$\langle$ a a a, b a a $\rangle \to$ 4/9

In general, on arbitrary *semirings*: $\langle \mathbb{A}, \oplus, \otimes, \mathbf{1}, \mathbf{0} \rangle$

A Note on *Weights*



$\langle$ a a a, b b b $\rangle \to$ 1⁄9

$\langle$ a a a, b a b $\rangle \to$ 2⁄9

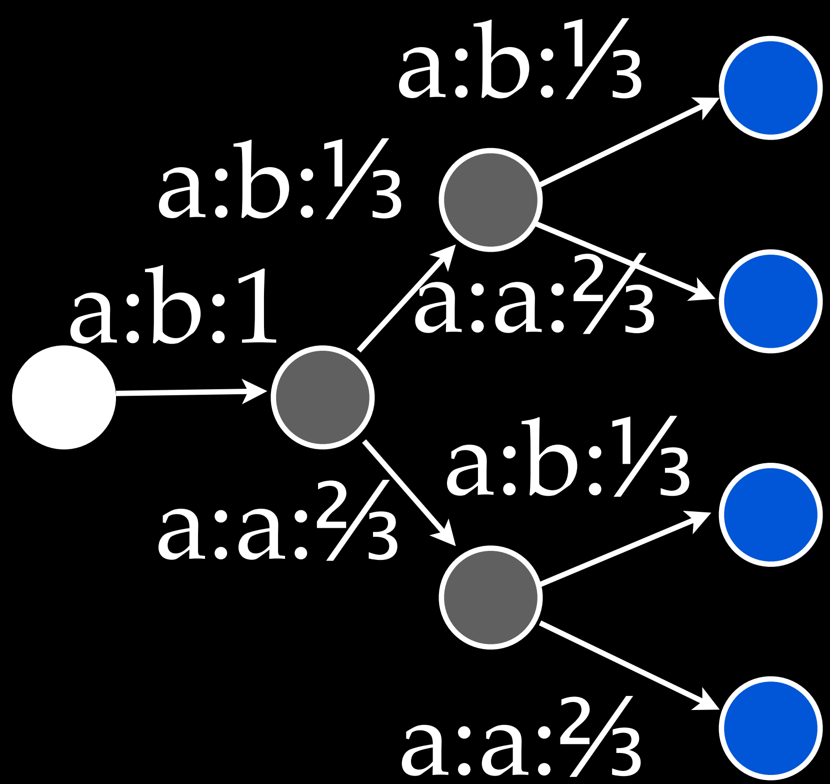
$\langle$ a a a, b b a $\rangle \to$ 2⁄9

$\langle$ a a a, b a a $\rangle \to$ 4⁄9

Where does 1⁄9 come from?

In general, on arbitrary *semirings*: $\langle \mathbb{A}, \oplus, \otimes, \mathbf{1}, \mathbf{0} \rangle$

A Note on *Weights*



$\langle$ a a a, b b b $\rangle \rightarrow$ 1⁄9

$\langle$ a a a, b a b $\rangle \rightarrow$ 2⁄9

$\langle$ a a a, b b a $\rangle \rightarrow$ 2⁄9

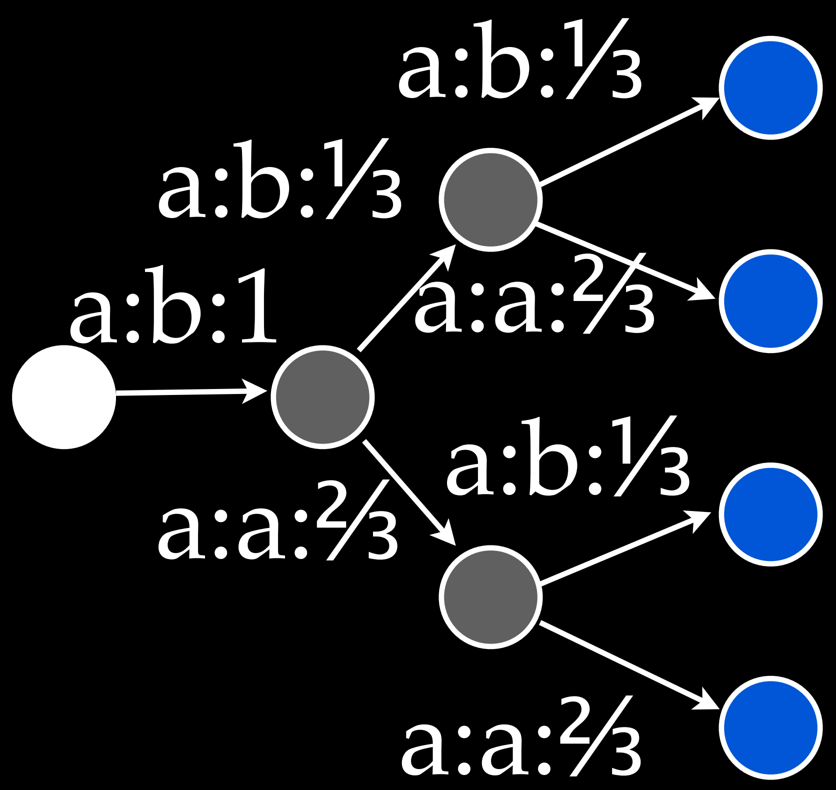
$\langle$ a a a, b a a $\rangle \rightarrow$ 4⁄9

Where does 1⁄9 come from?

A: $1 \times 1/3 \times 1/3$

In general, on arbitrary *semirings*: $\langle \mathbb{A}, \oplus, \otimes, \mathbf{1}, \mathbf{0} \rangle$

A Note on *Weights*



$\langle$ a a a, b b b $\rangle \rightarrow$ 1/9

$\langle$ a a a, b a b $\rangle \rightarrow$ 2/9

$\langle$ a a a, b b a $\rangle \rightarrow$ 2/9

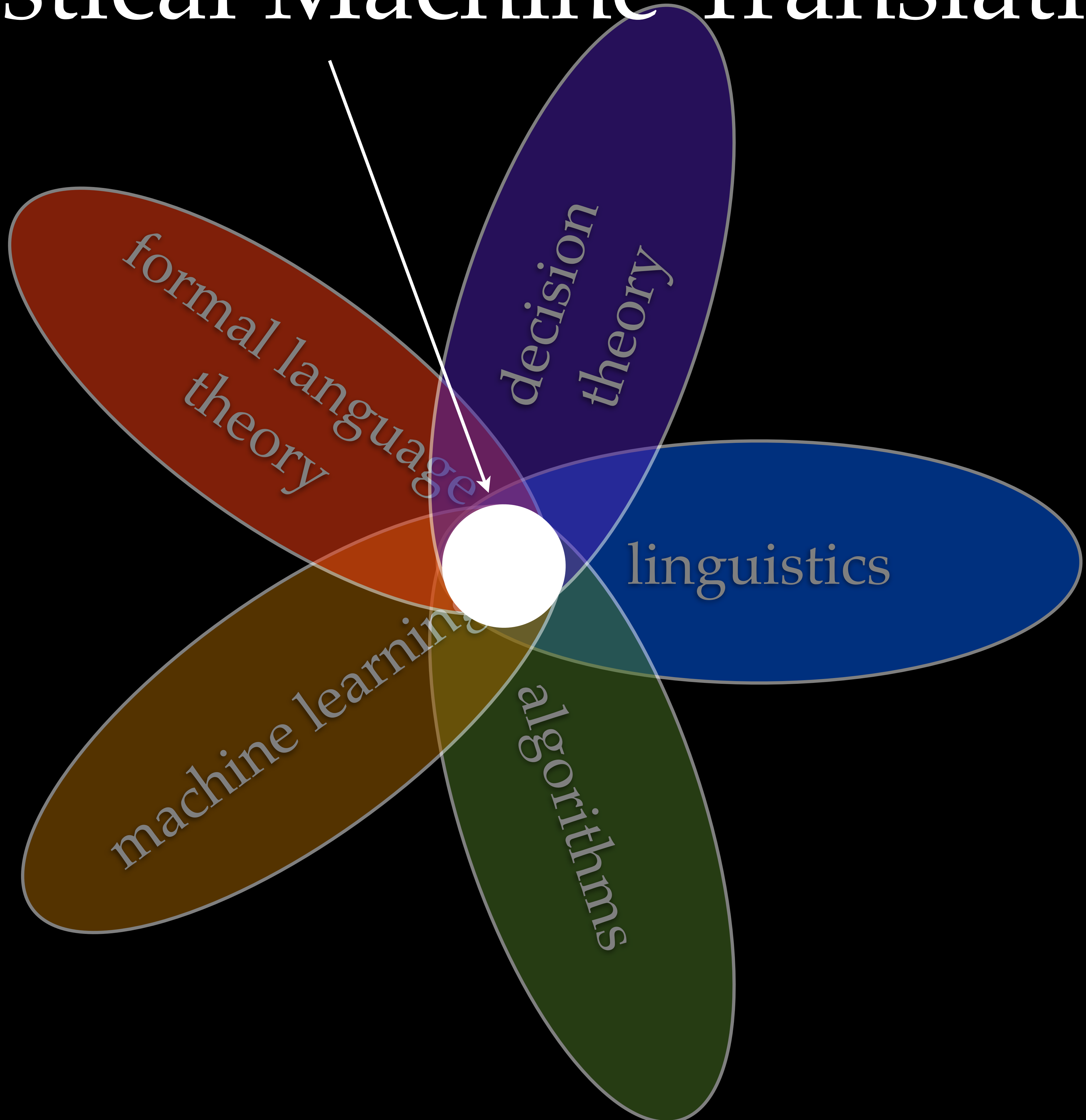
$\langle$ a a a, b a a $\rangle \rightarrow$ 4/9

Where does 1/9 come from? A: $1 \times 1/3 \times 1/3$

In the log domain: $\log(1) + \log(1/3) + \log(1/3)$

In general, on arbitrary *semirings*: $\langle \mathbb{A}, \oplus, \otimes, \mathbf{1}, \mathbf{0} \rangle$

Statistical Machine Translation



Next Week:

Non-finite-state models of translation!