University of Copenhagen
Department of Computer Science
PhD defence, 17 November 2026

ellis

PIONEER CENTRE FOR ARTIFICIAL INTELLIGENCE

Transducing Language Models

Vésteinn Snæbjarnarson

Advisors

Serge Belongie
University of Copenhagen

Ryan Cotterell
ETH Zürich

Committee

David Chiang
University of Notre Dame

Mirella Lapata
University of Edinburgh

Anders Søgaard
University of Copenhagen

Slides and thesis

vesteinn.is/tlm

Language models are remarkable

Answering questions

What is the capital of France?

Paris.

Writing code

Area of a circle, in Python?

```
def area(r):
    return pi * r**2
```

Translating

Translate to English: en kat

a cat

Drafting emails

Thank Serge for his help.

Dear Serge,
thank you for your help.

“Solving” Navier–Stokes

Can Navier–Stokes blow up?

Yes. Here is a proof in Lean.

Explaining

Why is the sky blue?

Air scatters blue light the most.

Problem example I

Ranking language models

Glucose is transported into the muscle cell:
A. via protein transporters called GLUT4.
B. only in the presence of insulin.
C. via hexokinase.
D. via monocarboxylic acid transporters.

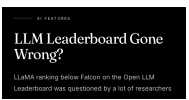
Question from Fourrier et al. (2023)



generallyintelligent.substack.com



huggingface.co/blog



analyticsindiamag.com

Fourrier et al. (2023); Holtzman et al. (2021); Wang et al. (2024)

Problem example I

Ranking language models

Glucose is transported into the muscle cell:
A. via protein transporters called GLUT4.
B. only in the presence of insulin.
C. via hexokinase.
D. via monocarboxylic acid transporters.

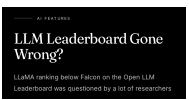
Question from Fourrier et al. (2023)



generallyintelligent.substack.com



huggingface.co/blog



analyticsindiamag.com

Three ways to produce an answer

1. the most likely letter
2. what the model writes
3. the most likely full answer

Fourrier et al. (2023); Holtzman et al. (2021); Wang et al. (2024)

Problem example I

Ranking language models

Glucose is transported into the muscle cell:
A. via protein transporters called GLUT4.
B. only in the presence of insulin.
C. via hexokinase.
D. via monocarboxylic acid transporters.

Question from Fourrier et al. (2023)



generallyintelligent.substack.com



huggingface.co/blog



analyticsindiamag.com

Three ways to produce an answer

1. the most likely letter
2. what the model writes
3. the most likely full answer

Many ways to phrase one answer

A (A) The answer is A. GLUT4

via protein transporters called GLUT4

Fourrier et al. (2023); Holtzman et al. (2021); Wang et al. (2024)

Problem example I

Ranking language models

Glucose is transported into the muscle cell:
A. via protein transporters called GLUT4.
B. only in the presence of insulin.
C. via hexokinase.
D. via monocarboxylic acid transporters.

Question from Fourrier et al. (2023)



generallyintelligent.substack.com



huggingfacе.co/blog



analyticsindiamag.com

Three ways to produce an answer

1. the most likely letter
2. what the model writes
3. the most likely full answer

Many ways to phrase one answer

A | (A) | The answer is A. | GLUT4 | via protein transporters called GLUT4

The three approaches rank **differently**.

Fourrier et al. (2023); Holtzman et al. (2021); Wang et al. (2024)

Problematic example II

Tasks impacted by tokenization

Autocomplete

The capital of France is ␣Paris.

The capital of France is␣ **1,000 years old this year.**

what the model reads

`is Paris` ⟶ `␣is` `␣Paris`

`is␣` ⟶ `␣is` `␣`

`is 1,000` ⟶ `␣is` `␣` `1` `,` `000`

Llama 3.2 (3B); cf. token healing (Guidance); Cao & Rimell (2021); Geh et al. (2024)

Problematic example II

Tasks impacted by tokenization

Autocomplete

The capital of France is ␣Paris.

The capital of France is␣ **1,000 years old this year.**

what the model reads

`is Paris` ⟶ `␣is` `␣Paris`

`is␣` ⟶ `␣is` `␣`

`is 1,000` ⟶ `␣is` `␣` `1` `,` `000`

Error correction

Fix the typo: please take two **tabkets** after every meal.

Please take two **tickets** after every meal.

what the model reads

`tablets` ⟶ `␣tablets`

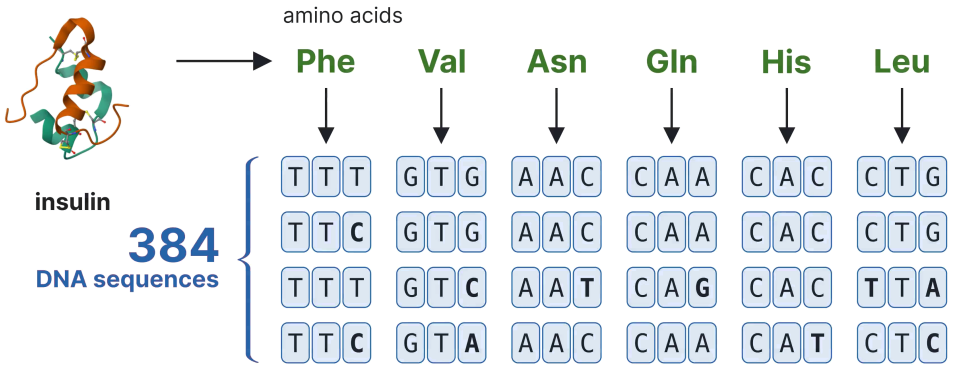
`tabkets` ⟶ `␣tab` `k` `ets`

`tickets` ⟶ `␣tickets`

Llama 3.2 (3B); cf. token healing (Guidance); Cao & Rimell (2021); Geh et al. (2024)

Problematic example III

Relating a protein to its DNA



We probably want to take all of them into account.

Structure: RCSB PDB 3I40

Three challenges

I Marginalization

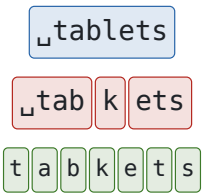
A (A)

The answer is A.

GLUT4

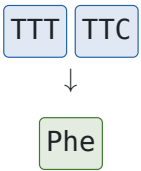
The unit is a class of equivalent answers.

II Boundaries



Granularity fitting the task.

III Domain



A new alphabet

Three challenges

I Marginalization

A (A)

The answer is A.

GLUT4

The unit is a class of equivalent answers.

II Boundaries

␣tablets

␣tab k ets

t a b k e t s

Granularity fitting the task.

III Domain

TTT TTC

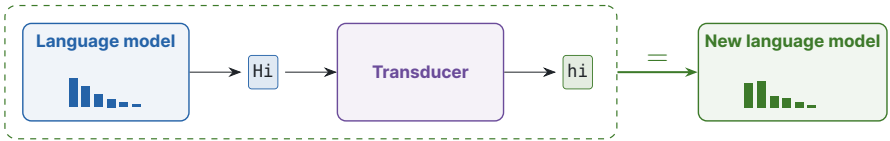
↓

Phe

A new alphabet

The units a model is defined over can get in the way of how we want to use it.

This thesis



Part I	**Transducing language models**	the diagram above	ICLR 2026
Part II	**Stochastic estimation of TLMs**	unbiased estimates	ICLR 2027 subm.
Part III	**Units in psycholinguistics**	units reading time studies	ACL 2026

Also first-authored during the PhD (full list in thesis):

- Causally Evaluating the Learnability of Formal Language Tasks (ICML 2026)
- Taxonomy-Aware Evaluation of Vision-Language Models (CVPR 2025)
- Discriminative Class Tokens for Text-to-Image Diffusion Models (ICCV 2023)
- Transfer to a Low-Resource Language via Close Relatives: The Case Study on Faroese (NoDaLiDa 2023)

Part I

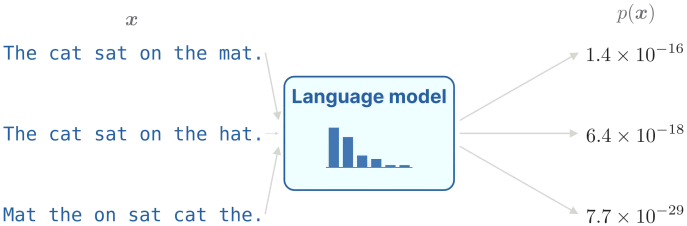
Transducing Language Models

ICLR 2026

Vésteinn Snæbjarnarson*, Samuel Kiegeland*, Tianyu Liu, Reda Boumasmoud, Ryan Cotterell, Tim Vieira

* equal contribution

A language model scores strings

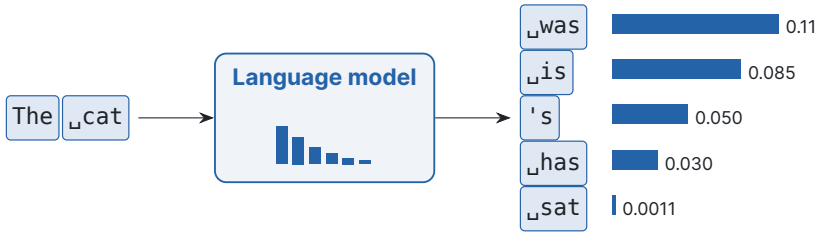


A probability for every string of tokens.

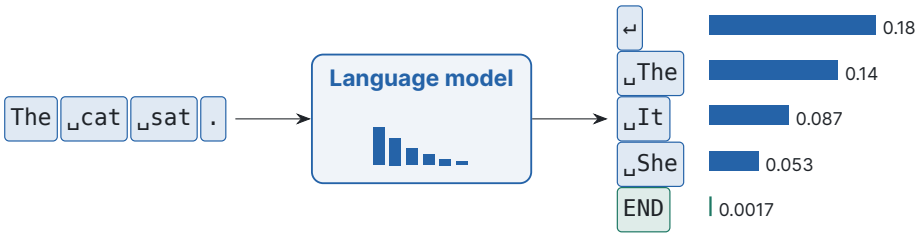
$$\sum_{\boldsymbol{x} \in \Sigma^*} p(\boldsymbol{x}) = 1$$

The probabilities of all strings sum to one.

The language model interface



The language model interface



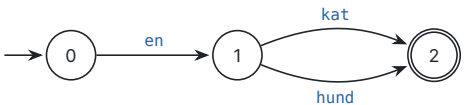
$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t \mid \boldsymbol{x}_{<t}) \cdot p(\text{end} \mid \boldsymbol{x})$$

Language models score strings **autoregressively**.

GPT-2 (Radford et al., 2019)

Transducers as string-to-string functions

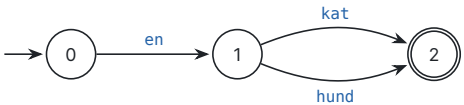
A state machine accepts a set of strings



en kat	✓ **accepted**
kat en	× **rejected**

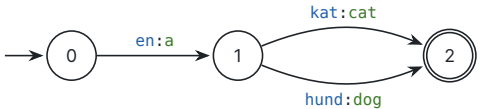
Transducers as string-to-string functions

A state machine accepts a set of strings



en kat	✓ accepted
kat en	× rejected

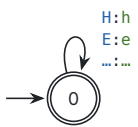
A transducer adds an output to each transition



en kat	⟶	a cat
en hund	⟶	a dog

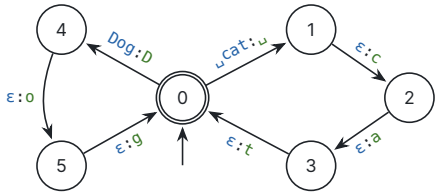
Transducers encoding functions

Lowercasing



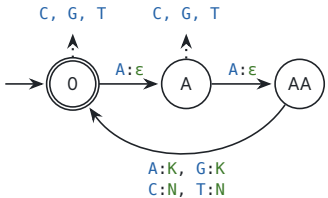
Hello ↦ hello

Tokens to characters



␣cat ↦ ␣cat Dog ↦ Dog

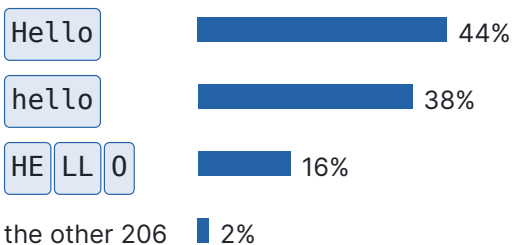
DNA to amino acids



AAA ↦ K AAC ↦ N

209 GPT-2 token strings lowercase to hello

Share of the probability



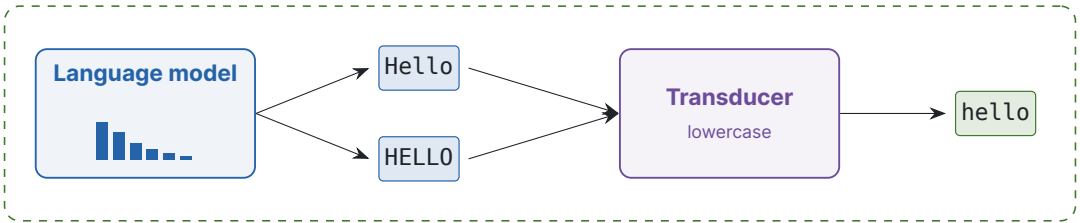
The other 206

hell o · Hell o · H ello · h ello · he llo
He llo · hell O · he LL O · Hell O · H ell o
hel lo · H ELL O · he ll o · HE LL o · Hel lo
He ll o · He LL O · h ELL O · H ell O · h ell o
he l lo · HE llo · He l lo · h E llo · H E llo
Hel Lo · H E LL O · He ll O · he ll O · H EL Lo
…

$$p_Y(\texttt{hello}) = \sum_{\boldsymbol{x}\,:\,f(\boldsymbol{x})=\texttt{hello}} p_X(\boldsymbol{x}) \approx 3.0 \times 10^{-8}$$

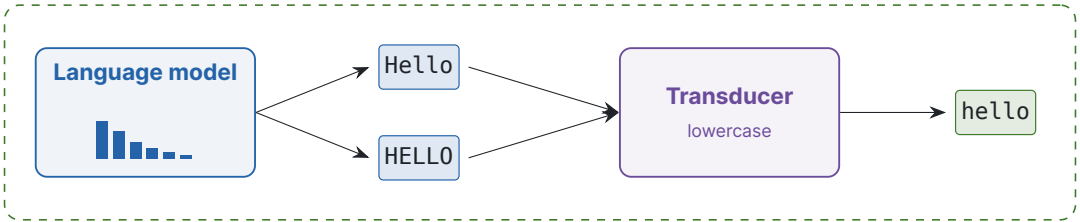
GPT-2 small; probabilities include END

Proposal: Combine a language model with a transducer



a new language model, over target strings

Proposal: Combine a language model with a transducer



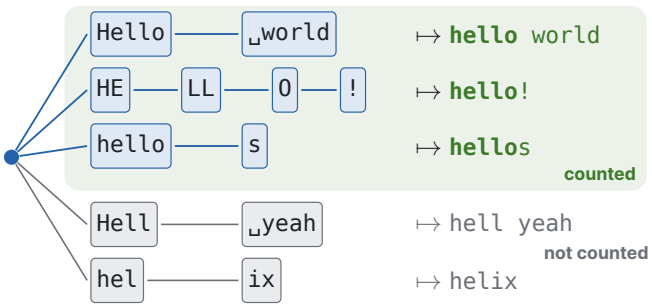
a new language model, over target strings

$$X \sim p_X \implies f(X) \sim p_Y \qquad p_Y(\boldsymbol{y}) = \sum_{\boldsymbol{x}\,:\,f(\boldsymbol{x})=\boldsymbol{y}} p_X(\boldsymbol{x})$$

A prefix probability

The probability that the output **starts with** `hello`

$$\overrightarrow{p}_Y(\texttt{hello}) = \Pr\left[\,\texttt{hello} \preceq f(X)\,\right]$$



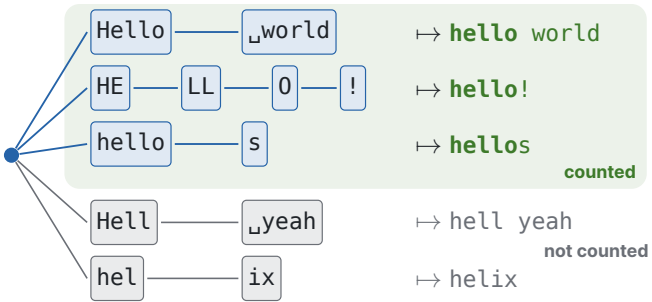
The ratio of two prefix probabilities is the next-symbol probability: the interface we are used to.

$$p_Y(\texttt{o} \mid \texttt{hell}) = \frac{\overrightarrow{p}_Y(\texttt{hello})}{\overrightarrow{p}_Y(\texttt{hell})}$$

A prefix probability

The probability that the output **starts with** `hello`

$$\overrightarrow{p}_Y(\texttt{hello}) = \Pr\left[\texttt{hello} \preceq f(X)\right]$$



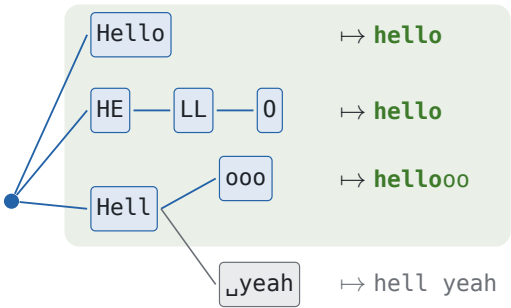
The ratio of two prefix probabilities is the next-symbol probability: the interface we are used to.

$$p_Y(\texttt{o} \mid \texttt{hell}) = \frac{\overrightarrow{p}_Y(\texttt{hello})}{\overrightarrow{p}_Y(\texttt{hell})}$$

In contrast, a string probability needs the string to end: $p_Y(\texttt{hello}) = \Pr\left[f(X) = \texttt{hello}\right]$, with `END`.

How do we get the prefix probabilities?

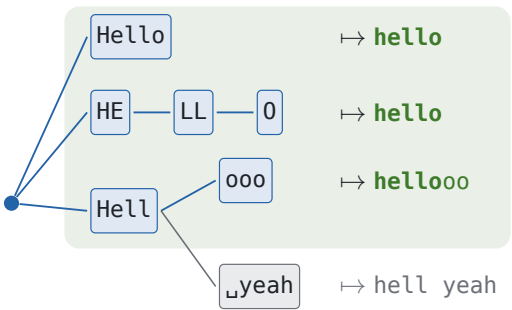
Issue 1: a token can run past the target



`Hell` `ooo` never spells `hello` exactly, yet its output starts with it.

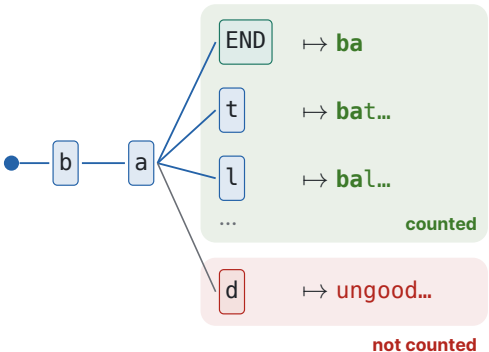
How do we get the prefix probabilities?

Issue 1: a token can run past the target



`Hell` `ooo` never spells `hello` exactly, yet its output starts with it.

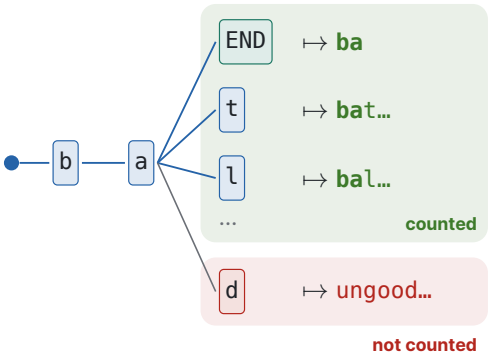
Issue 2: some transformations need to look ahead



Newspeak: `bad` ↦ `ungood`. After `ba`, the output is not yet decided.

Example: the target prefix ba

Which strings map to strings with ba?

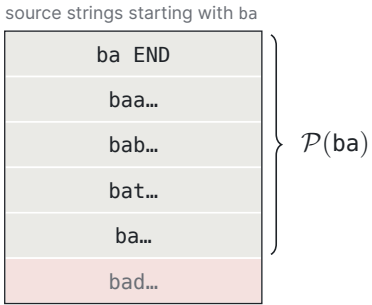


A machine accepts exactly these



State 2 accepts ba. From there anything but d leads to state 3, where every extension is accepted.

The precover of ba



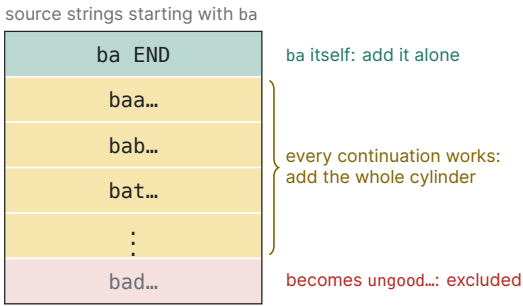
Every source string whose output starts with ba.

$$\mathcal{P}(\boldsymbol{y}) = \{\boldsymbol{x} : \boldsymbol{y} \preceq f(\boldsymbol{x})\}$$

$$\overrightarrow{p}_Y(\boldsymbol{y}) = \sum_{\boldsymbol{x} \in \mathcal{P}(\boldsymbol{y})} p_X(\boldsymbol{x})$$

Infinitely many source strings: we cannot sum them one by one.

One prefix represents infinitely many strings

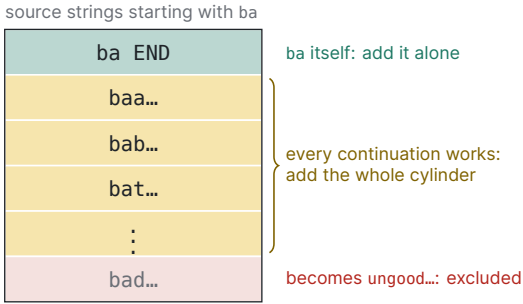


Every string that starts with baa

baa baaa baab baac …



One prefix represents infinitely many strings





Every string that starts with baa

baa baaa baab baac …

Summing them up is the prefix probability

$$p_X(\langle \text{baa} \rangle) = \sum_{\boldsymbol{x} \in \langle \text{baa} \rangle} p_X(\boldsymbol{x}) = \overrightarrow{p}_X(\text{baa})$$

This set is called the *cylinder* of baa.

Infinitely many strings,
$\implies$ finitely many prefixes.

Some states give prefix probabilities



state 2: accepting, not universal → the string ba

state 3: universal → whole cylinders

source strings starting with ba

ba END	ba itself: add it alone
baa…	every continuation works: add the whole cylinder
bab…	
bat…	
⋮	
bad…	becomes ungood…: excluded

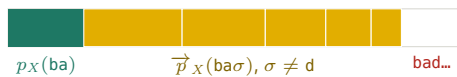
$$\mathcal{P}(\boldsymbol{y}) = \Big(\bigcup_{\boldsymbol{q} \in \mathcal{Q}(\boldsymbol{y})} \langle \boldsymbol{q} \rangle \Big) \cup \mathcal{R}(\boldsymbol{y})$$

$$\overrightarrow{p}_Y(\boldsymbol{y}) = \sum_{\boldsymbol{q} \in \mathcal{Q}(\boldsymbol{y})} \overrightarrow{p}_X(\boldsymbol{q}) + \sum_{\boldsymbol{r} \in \mathcal{R}(\boldsymbol{y})} p_X(\boldsymbol{r})$$

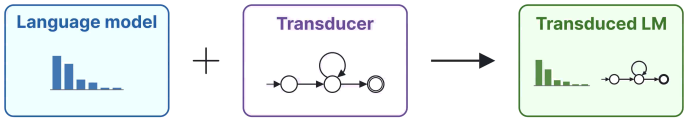
Quotient
whole cylinders, added at once

Remainder
single strings that end

accumulated probability for ba



Summary: composing a language model and a transducer gives a new language model



No retraining: source model used as is

Prefix probabilities give the full interface

Exact when Quotient–Remainder decomposition is finite

Part II

Stochastic Estimation of Transduced Language Models

ICLR 2027 under review

Vésteinn Snæbjarnarson, Samuel Kiegeland, Manuel de Prada Corral, Ryan Cotterell, Tim Vieira

Recap: exact, but many summands

$$\overrightarrow{p}_Y(\boldsymbol{y}) = \sum_{\boldsymbol{q} \in \mathcal{Q}(\boldsymbol{y})} \overrightarrow{p}_X(\boldsymbol{q}) + \sum_{\boldsymbol{r} \in \mathcal{R}(\boldsymbol{y})} p_X(\boldsymbol{r})$$

Exact when both sums are finite.

But the sums can have very many terms.

The quotient can be huge

DNA → amino acids ATT TTC AAA → I F K

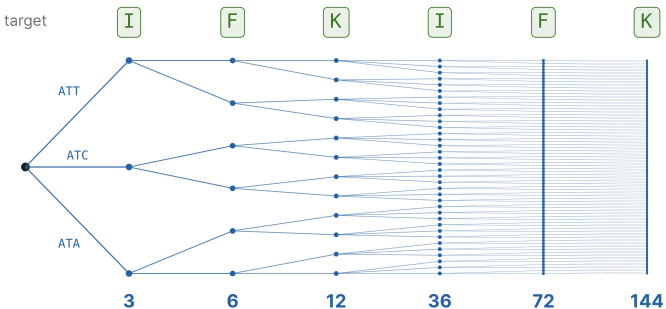
source prefixes for IFK: 3 × 2 × 2 = 12



$$(\texttt{IFK})^k \implies 12^k \text{ prefixes}$$

Part II

Every new amino acid multiplies the source prefixes



after 9 amino acids

1,728

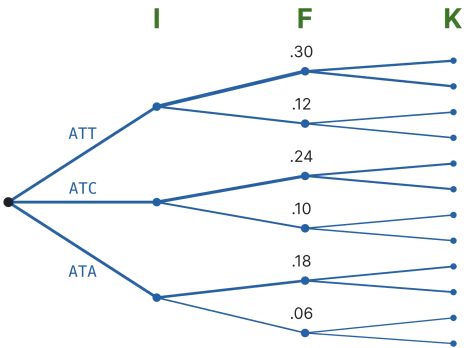
source prefixes

after 222 amino acids

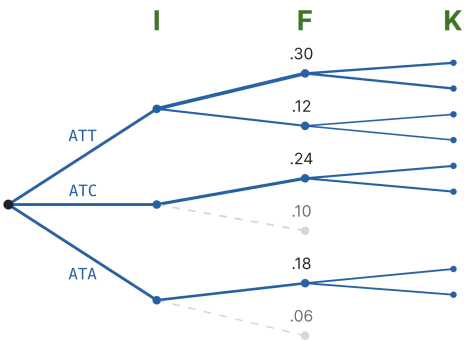
7.2×10^{79}

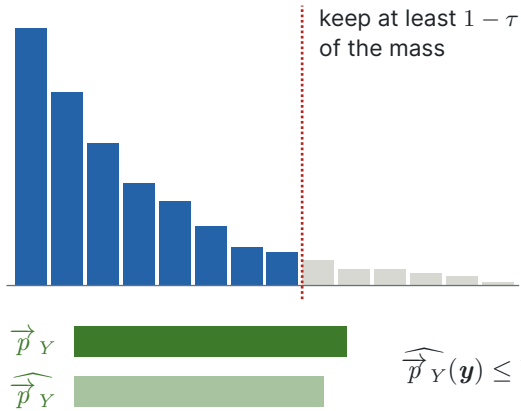
source prefixes

One solution: prune light branches

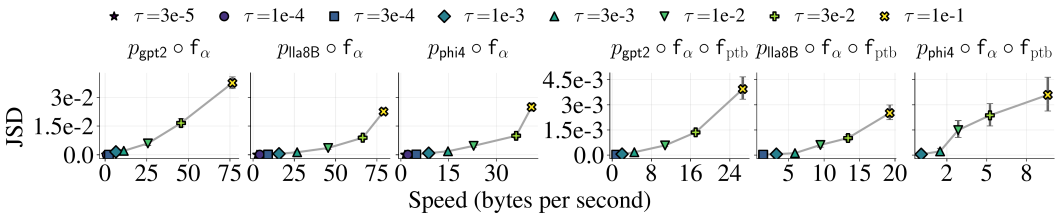


One solution: prune light branches





Tighter pruning is closer to the reference, but slower



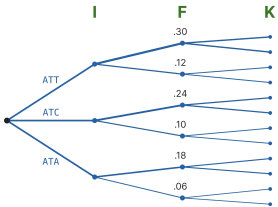
Each point corresponds to a pruning threshold τ. Lower means closer to the reference.

Part II

Which source prefixes do we extend?

First paper: Transducing Language Models

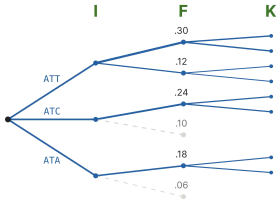
This paper



Exact: extend every prefix

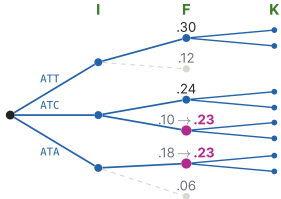
Nothing is lost.

The leaves multiply with every amino acid.



Pruning: only heaviest

Lightest prefixes are dropped.

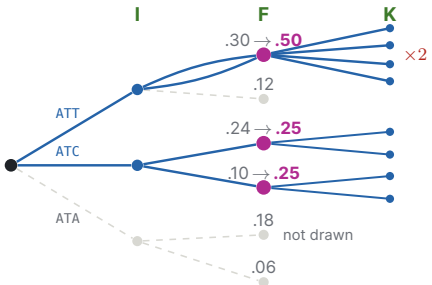


Sample and reweight: unbiased

Heavier prefixes are more likely to be kept. Lighter ones are sampled.

Sample, then reweight

Keep a prefix with probability π, count it as w/π: $\mathbb{E}\left[\mathbf{1}[\text{kept}] \cdot \frac{w}{\pi}\right] = \pi \cdot \frac{w}{\pi} = w$



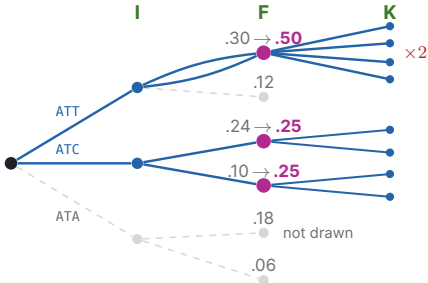
With replacement

A prefix drawn twice repeats the same computation.

Horvitz & Thompson (1952); Fearnhead & Clifford (2003); Shah & Kroese (2018)

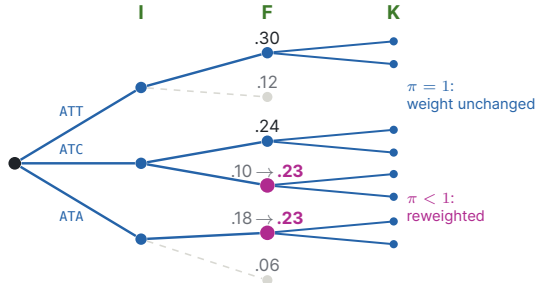
Sample, then reweight

Keep a prefix with probability π, count it as w/π: $\mathbb{E}\left[\mathbf{1}[\text{kept}] \cdot \frac{w}{\pi}\right] = \pi \cdot \frac{w}{\pi} = w$



With replacement

A prefix drawn twice repeats the same computation.

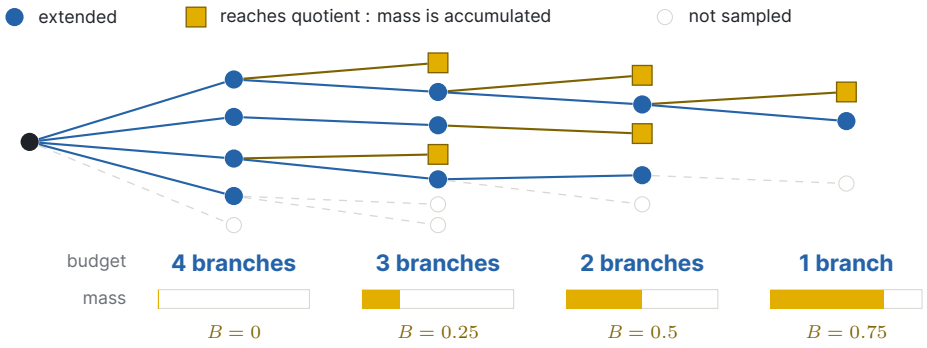


Without replacement: each prefix at most once

Every kept prefix counts as w/π. Heavy ones have $\pi = 1$.

Horvitz & Thompson (1952); Fearnhead & Clifford (2003); Shah & Kroese (2018)

The budget shrinks as mass accumulates

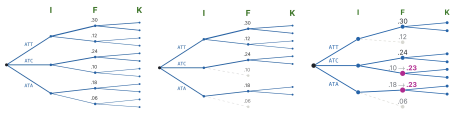


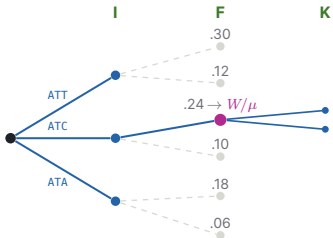
One particle per $\tau = \dfrac{B + W}{M}$ of mass: $\pi_i = \min\left(1,\ w_i/\tau\right)$, particles kept $\approx \dfrac{W}{\tau} = M \cdot \dfrac{W}{B + W}$

M: the largest budget; B: accumulated mass; W: mass still live.

If no mass is found, the budget is shrunk each step to ensure halting.

Flip a coin to ensure halting





sampling with roulette

If single branch is left, flip a coin.

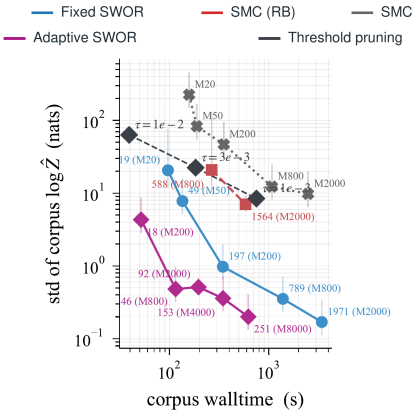
Continue with probability μ, and count the survivor as W/μ:

$$\mathbb{E}\left[\frac{\mathbf{1}[\text{continue}]}{\mu}\ W\right] = W$$

W: the mass still to be extended.
The estimate stays unbiased, and the run stops with probability one.

Kahn (1955); illustrative numbers

Adaptive sampling has the lowest variance for its cost

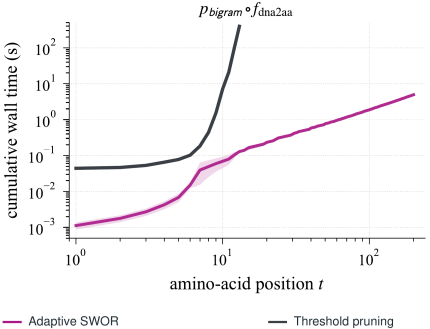


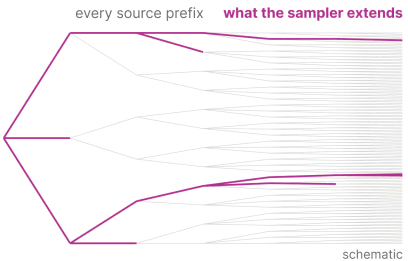
Adaptive sampling without replacement is lowest at every walltime.

At matched variability (0.36 vs 0.35 nats) it is 4× faster than the fixed-budget sampler.

Between-seed standard deviation is 49–57× below sampling with replacement.

Sampling follows a few branches of a huge tree





Threshold pruning: over 5 minutes at position 13. **Adaptive sampling**: position 200 in about 5 s.

Unbiased prefix probabilities; lower variance costs computation.

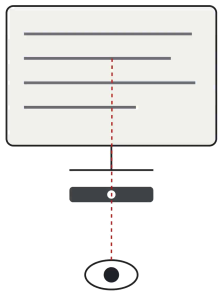
Part III

On the Proper Treatment of Units in Surprisal Theory

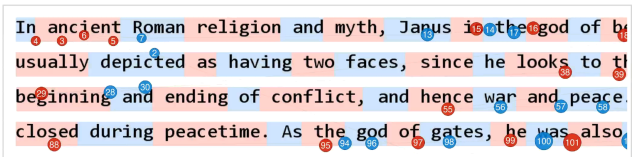
ACL 2026

Samuel Kiegeland, Vésteinn Snæbjarnarson, Tim Vieira, Ryan Cotterell

An eye-tracking study records where people look



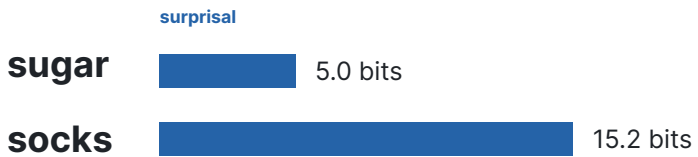
short texts are on a screen
while a camera tracks their eyes



Data: each circle a fixation

Surprisal: how unexpected the model finds a word

I take my coffee with cream and _____



$$S(u \mid \boldsymbol{h}) = -\log p(u \mid \boldsymbol{h})$$

Do people read the surprising word more slowly?

GPT-2 Small; Hale (2001); Levy (2008)

Does surprisal help predict reading times?

English MECO 46 readers 12 texts GPT-2 Small

predictors

word length

word frequency

previous word

reader

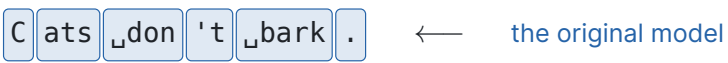
+ surprisal



Δ: how much better the model predicts unseen reading times with surprisal.

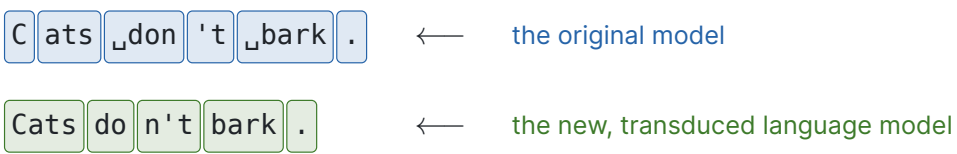
Transducing the model to fit words

Cats don’t bark.



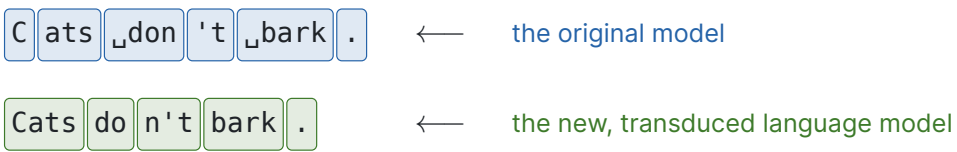
Transducing the model to fit words

Cats don’t bark.



Transducing the model to fit words

Cats don’t bark.



The framework lets us use the units the scientific question is about.

Prior work: split at spaces

One word becomes two different units

Hale cited Levy ⟶ `Hale` `␣cited` `␣Levy`

Levy cited Hale ⟶ `Levy` `␣cited` `␣Hale`

At the start of a string, `Hale` has no space before it. Elsewhere it does.

Prior work: split at spaces

One word becomes two different units

Hale cited Levy ⟶ `Hale` `␣cited` `␣Levy`

Levy cited Hale ⟶ `Levy` `␣cited` `␣Hale`

At the start of a string, `Hale` has no space before it. Elsewhere it does.

Transduced: `Hale` `cited` `Levy`

`Levy` `cited` `Hale`

One word, one unit, wherever it stands.

Prior work: split at spaces

One word becomes two different units

Hale cited Levy ⟶ `Hale` `␣cited` `␣Levy`

Levy cited Hale ⟶ `Levy` `␣cited` `␣Hale`

At the start of a string, `Hale` has no space before it. Elsewhere it does.

Transduced: `Hale` `cited` `Levy`

`Levy` `cited` `Hale`

One word, one unit, wherever it stands.

Where to split depends on the context

1,000 ⟶ `1,000`

end, he said ⟶ `end` `,` `he` `said`

The same comma is kept in one case and split off in the other. No fixed list of delimiters does both.

A four step proposal for reading-time studies

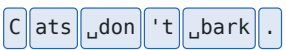
1 Choose the units

characters, tokens, words

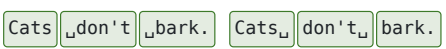
Characters



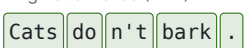
Model tokens



Orthographic words, leading or trailing space



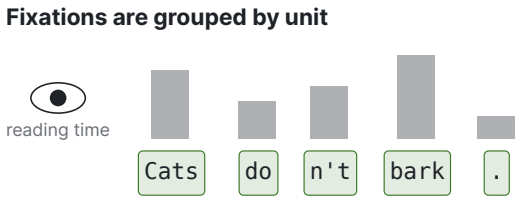
Linguistic words (PTB)



A four step proposal for reading-time studies

1 **Choose the units**
characters, tokens, words

2 **Segment the reading data**
reading times per unit



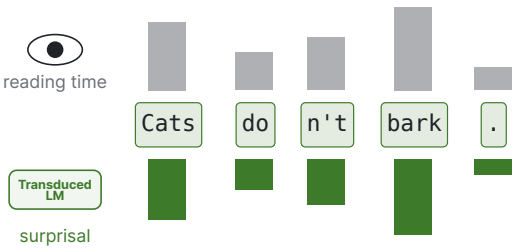
Part III

A four step proposal for reading-time studies

1 **Choose the units**
characters, tokens, words

2 **Segment the reading data**
reading times per unit

3 **Transduce the language model**
surprisal per unit



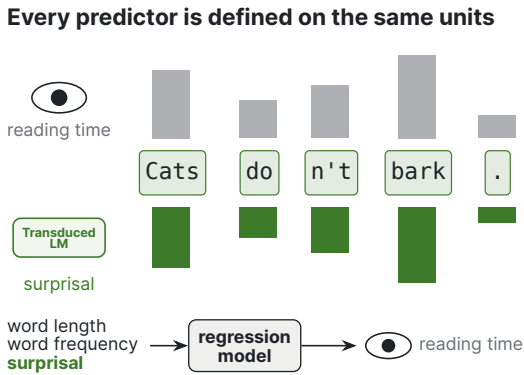
A four step proposal for reading-time studies

1 Choose the units
characters, tokens, words

2 Segment the reading data
reading times per unit

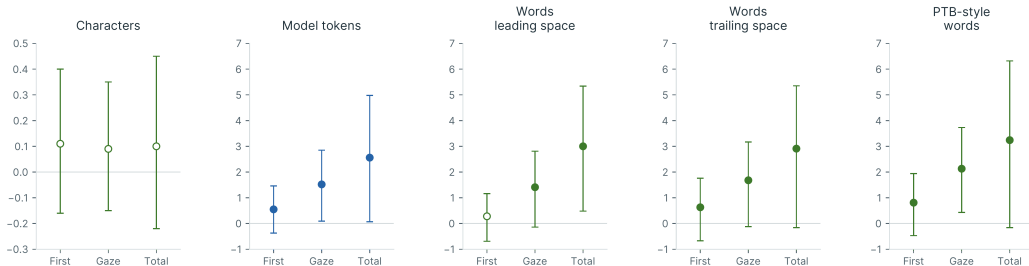
3 Transduce the language model
surprisal per unit

4 Keep the predictors consistent
length and frequency on the same units



Part III

The unit choice changes the prediction task



Improvement in held-out prediction (10^{-3} nats / observation); 95% trial-bootstrap intervals

Filled markers: paired permutation test $p < .05$. Character panel uses a different scale.

Surprisal improves prediction under tokens and word-like units, not under characters.

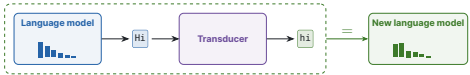
The researcher should state their units.

Contributions

Part I

Transducing language models

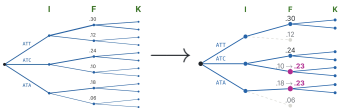
A language model composed with a transducer is a new language model, exact when the decomposition is finite.



Part II

Stochastic estimation of TLMs

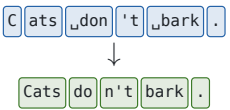
Sampling replaces pruning: unbiased prefix probabilities, with a budget that shrinks as mass accumulates.



Part III

Units in psycholinguistics

The researcher states the units and the model is transduced to them: the same units for reading times and every predictor.



Limitations

Speed

GPT-2 Small

2,200

characters per second

transduced to PTB words

12

characters per second

About 180 times slower.

Fine for a reading-time corpus.
Slow for commercial LLMs.

The transducer has to be written

The framework takes the transducer as given.

Example I

A (A) The answer is A. GLUT4

A transducer from every phrasing to its answer is still open.

Future work

Scientific applications
scoring proteins with DNA models, molecules across notations

Combining models across vocabularies
distillation, ensembling, speculative decoding

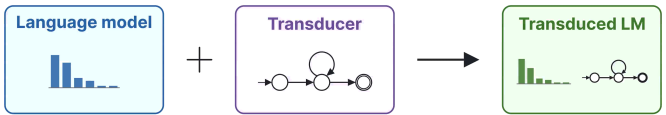
Stochastic transducers
one source string, several weighted outputs

Training through the transducer
fit the source model directly to data in the target units

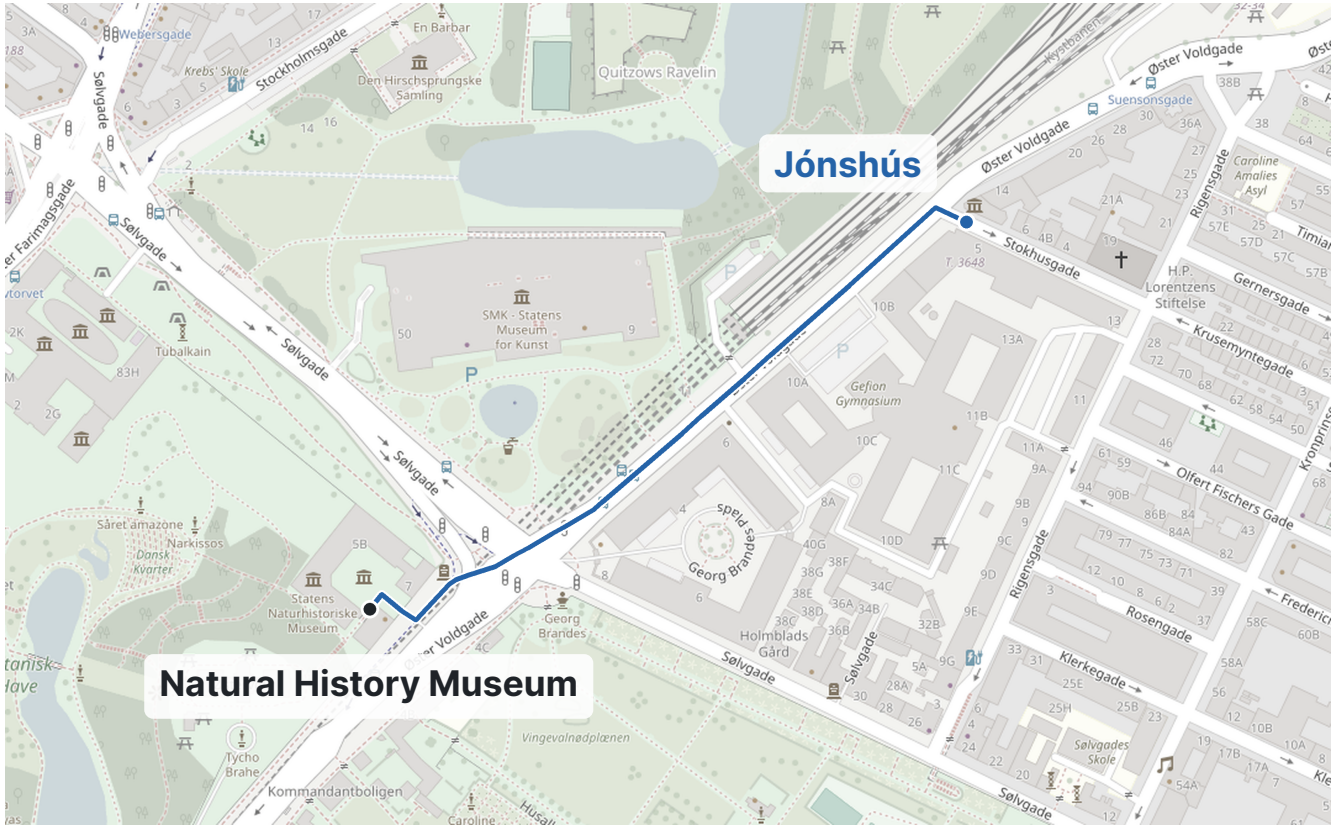
When is the decomposition finite?
necessary conditions are open

Sampling the next-symbol distribution
the sampler scores a given target only

Thank you!



Reception at Jónshús



After the defence

Jónshús
the Icelandic Culture House

Øster Voldgade 12, 3rd floor

The historic apartment of Jón Sigurðsson and Ingibjörg Einarsdóttir

400 m along Øster Voldgade, about 5 minutes on foot

The Icelandic flag hangs outside.

Map: © OpenStreetMap contributors

Extra slides

X.2 Normalization of the transduced model
X.3 What it means to cover a target
X.4 The size of the quotient
X.5 Finite decompositions: sufficient conditions
X.6 The safety example
X.7 The Newspeak machine
X.8 Computational cost
X.9 The twist
X.10 Inclusion probabilities
X.11 Unbiased prefix probabilities, biased surprisal
X.12 Comparing inventories on common regions
X.13 Whitespace-based corrections compute a different event
X.14 Zero surprisals in the character analysis
X.15 Characters as units of fixation
X.16 Pruning and sampling lead to the same conclusions
X.17 Scope of the reading-time study
X.18 Stochastic unit parsers
X.19 The decomposition algorithm
X.20 Frontier example and lazy determinization
X.21 Beam summing for transduced language models
X.22 Sampling without replacement
X.23 Systematic sampling
X.24 The smc_rb algorithm
X.25 The cost is linear in the size of the machine

Normalization of the transduced model

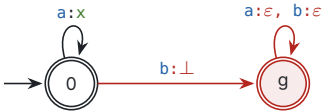
A partial map loses mass. A garbage state makes it total.

partial



aab has no output

made total



aab ↦ xx⊥

Thesis tightness, Section 2.2 (p. 12) | total map, Section 3.1 (p. 18) | total extensions of the examples (pp. 28, 44) | halting, Appendix B.1.4 (p. 140) | erratum 2

What it means to cover a target

An output string covers the target when it starts with it.

Lowercasing, target `he`

`HE` `llo`	⟶	`hello`	✓ covers
`he`	⟶	`he`	✓ covers
`H`	⟶	`h`	× not yet
`Hi`	⟶	`hi`	× never

A source string covers y when its output does: $y \preceq f(x)$.

The precover $\mathcal{P}(y)$ is the set of all covering source strings; its mass is the prefix probability.

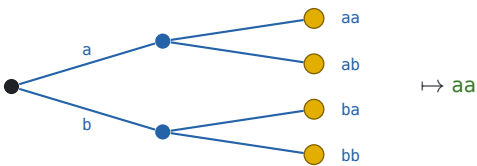
During the search, a buffer is the output written so far along one path. A covering buffer shows membership only when the path also ends in an accepting state.

Thesis precover, Eq. 3.3 (p. 19) | first use (p. 20) | buffers, Section 4.3.1 (p. 42) | erratum 3

The size of the quotient

At most k^m quotient elements, and the bound is attained.

Every source symbol maps to a; target aa



$\mathcal{Q}(\mathtt{aa}) = \{\mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}\}$: $k^m = 2^2$ elements

A string has one output. It either stops exactly at the target or continues past it, so a quotient element has only one kind of successor.

The thesis prints the looser bound $(k+1)^m$, which counts both kinds.

The proposition is due to Vieira et al.; the proof adapts their argument.

Thesis Proposition 3.2 (p. 24) | erratum 5

Finite decompositions: sufficient conditions

Lemma 3.1 gives sufficient conditions. They are not necessary.



$\mathcal{Q}(\varepsilon) = \{\varepsilon\}$
finite



$\mathcal{Q}(\mathtt{c}) = \{\mathtt{b}, \mathtt{ab}, \mathtt{aab}, \dots\}$
infinite

Both machines have a loop that reads a symbol and writes nothing. The lemma excludes such loops.

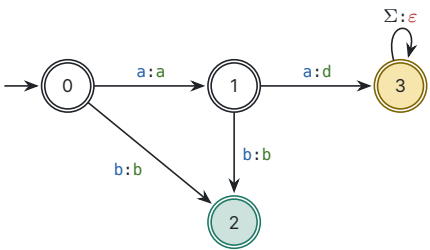
The first machine still has a finite decomposition for every target; the second does not.

A characterization of when decompositions are finite is open.

Thesis Lemma 3.1 and its proof (pp. 26–27) | an infinite quotient (p. 74) | open question, Section 7.3 (p. 107)

The safety example

Three ways a state can be safe, on one machine.



target a: $\mathcal{Q} = \{\mathtt{aa}\}, \quad \mathcal{R} = \{\mathtt{a}, \mathtt{ab}\}$

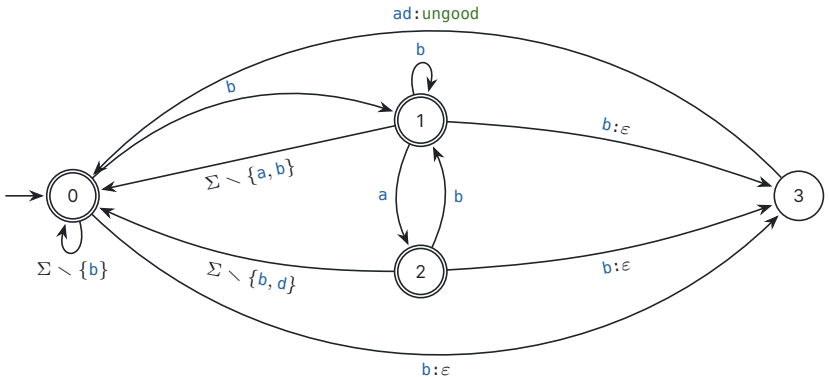
State 3 accepts every continuation. State 2 has no way on. States 0 and 1 are safe because all their successors are.

As printed, the loop at state 3 writes nothing, which breaks condition (i) of the lemma, and the function is partial.

With a loop that copies its input, condition (i) holds and the quotient and remainder are unchanged.

Thesis Example 3.4 (pp. 27–28) | Lemma 3.1 (p. 26) | erratum 6

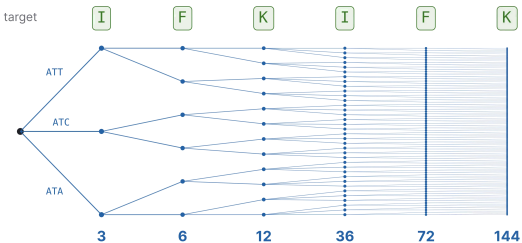
The Newspeak machine



Thesis Example 3.2 (p. 20) | the shorthand is stated for the compiled machines (Appendix C, p. 156) | erratum 4

Computational cost

The algorithm lists the quotient and remainder, so it costs at least their size.



How large they are depends on the transducer. Here the source prefixes multiply with every amino acid.

Only the part of the machine that the target prefix needs is determinized.

GPT-2 Small: about 2,200 characters per second. Transduced to PTB words: about 12.

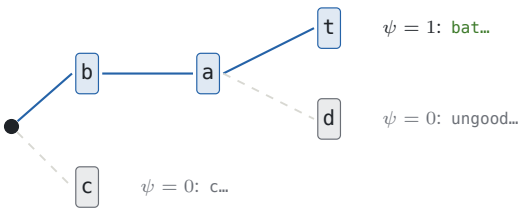
DNA to amino acids

Thesis lazy determinization, Section 4.3 (p. 42) | cylinder check (p. 47) | linear time (p. 49) | worst case, Appendix B.4 (p. 148) | 12.0 symbols/s, Table C.6 (p. 162)

The twist

A weight on source prefixes that says which ones are worth extending.

Newspeak, target ba: the live indicator



Live twist: $\psi(\boldsymbol{x}) = 1$ if some completion of $\boldsymbol{x}$ can still give an output starting with the target, and 0 otherwise. Every experiment uses it.

Optimal twist: the probability that a completion covers the target,

$$\psi^*(\boldsymbol{x}) = \Pr\left[\, \boldsymbol{y} \preceq f(X) \mid \boldsymbol{x} \preceq X \,\right]$$

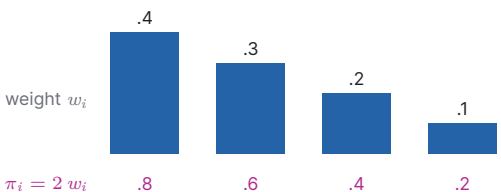
With it the estimate has zero variance. $\psi^*(\varepsilon)$ is the prefix probability itself.

Thesis introduced, Section 5.1.1 (p. 64) | optimal twist, Eq. 5.3 (p. 66) | live twist in every experiment (p. 78) | erratum 8

Inclusion probabilities

π_i is the probability that particle i is kept. It is a design choice.

Four particles, keep $M = 2$



A kept particle counts as $w_i/\pi_i = .5$.
Two kept: total 1.0, as before.

$\pi_i = \min(1,\ c\,w_i)$, with c set so that $\sum_i \pi_i = M$.

Any positive values give an unbiased sum. Proportional to weight makes every sampled survivor count the same, so the total is preserved exactly.

Dividing by π_i makes each particle count its own weight on average:

$$\mathbb{E}\left[\mathbf{1}\{i \text{ kept}\}\,\frac{w_i}{\pi_i}\right] = \pi_i \cdot \frac{w_i}{\pi_i} = w_i$$

Thesis term first used (p. 63) | unbiased sum, Eq. 5.9 (p. 68) | our choice, Eq. 5.11 (p. 68) | Algorithm 5.2 (p. 69) | erratum 7

Unbiased prefix probabilities, biased surprisal

The guarantee holds for the prefix probability, not for its logarithm or for ratios.

$\mathbb{E}[\widehat{Z}] = Z,$ but $\mathbb{E}[-\log \widehat{Z}] \geq -\log Z$ (Jensen: $-\log$ is convex)

Corpus surprisal, PTB words on MECO (2,264 units)

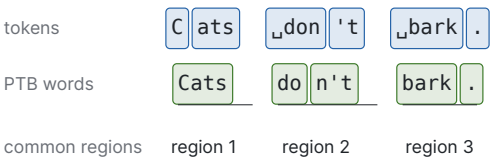
Pruning	7,486 nats	
Sampling without replacement	7,380 nats	
Difference	**106 nats**	**1.4%**, about 0.05 nats per unit

Bias from the logarithm, by jackknife over the ten runs: 1.55 nats (0.02%).

Thesis 106 nats, Section 6.5 (p. 103) | jackknife, Appendix C.6 (p. 164) | erratum 11

Comparing inventories on common regions

Each inventory is fitted on its own observations.



In the experiments each unit is its own region: 6 token observations against 5 word observations here; 33,514 to 53,614 over the corpus.

With common regions every inventory is fitted on the same 3 observations, so the fits are comparable. That analysis was not run.

Thesis observations per inventory, Table C.2 (p. 156) | regions, Section 6.2 (p. 93) | “not directly comparable” (p. 101)

Part III

Whitespace-based corrections compute a different event

A word needs its end. Without it, the prefix also counts longer words.

Two units, a and ab, each with probability $\frac{1}{2}$

starts with a — a | ab — probability 1

the unit a, ended — a — probability $\frac{1}{2}$

$a\,\Sigma^* \neq a\#\,\Sigma^*$, # marks the end of the unit

The transduced model scores the second event: the separator comes after the unit.

How far the two differ on real text was not measured.

Thesis critique of the whitespace-based corrections, Section 6.1.1 (pp. 89–90) | separator, Example 6.3 (p. 92)

Zero surprisals in the character analysis

The excluded zeros are near-certain word completions.

In anci**ent** Roman religion and myth, Janus is the god of beginni**ng**s and gates. He has a double nature and is usua**lly** depi**ct**e**d** as havi**ng** two faces, since he looks to the futu**r**e and to the past. Janus presided ov**er** the begi**n**ni**ng** and ending of confli**ct**

MECO, first text. **Red**: character surprisal exactly zero.

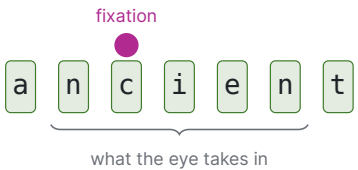
The beam keeps five candidates and rounds “almost certain” to “certain”. Excluded: 4,780 of 53,614 observations (8.9%).

Zero is a legitimate prediction there, so keeping them is also defensible; that variant was not run. The sampler reanalysis covers PTB words only.

Thesis exclusion, Section 6.4.2 (p. 100) | beam settings, Table C.5 (p. 161) | counts, Table C.7 (p. 165) | erratum 10

Characters as units of fixation

A fixation takes in more than the letter it lands on.



The reading time is credited to c alone.

The data records which letter each fixation lands on, and the reading time is credited to that letter alone.

The eye takes in its neighbours too. The unit and the region read do not match, which is a likely reason that surprisal shows no effect under characters.

Character surprisal suits regions below the word, such as punctuation or word beginnings. The study did not use such regions.

Thesis character inventory, Section 6.4.1 (p. 97) | fixations to characters, Section 6.4.2 (p. 100)

Pruning and sampling lead to the same conclusions

With this data the two estimators could not be separated.

Gain in held-out prediction from surprisal

	first fixation	gaze	total
Pruning	0.81	2.13	3.24
Sampling	0.76	2.05	3.26

Δ_{llh} in 10^{-3} nats per observation. PTB words, 33,472 observations.

With either estimator, surprisal improves prediction on all three measures, by nearly the same amounts.

A test between the two fits finds no difference: $p \geq .068$ in both directions, for each measure.

This is not evidence that they are equivalent: 12 texts, one corpus.

Thesis Section 6.5 (pp. 102–103) | Table 6.1 (p. 103) | details, Appendix C.6 (p. 164) | erratum 12

Scope of the reading-time study

One model and one corpus: a matter of data and cost.

Used	MECO, English: 46 readers, 12 texts; GPT-2 Small
Now available	
English	CELER, Dundee, EMTeC, OneStop, Provo, UCL
German	PoTeC
Danish	CopCo
Chinese	Beijing Sentence Corpus

Corpora with raw fixations.

The method re-assigns raw fixations to each choice of unit; most corpora publish only word-level reading times.

GPT-2 Small follows previous work, and the transduced model is slow: the frequency estimate alone took days of GPU time.

The claim is methodological and does not depend on the size of the study.

Thesis limitations, Section 6.6 (pp. 103–104) | model, Section 6.4.3 (p. 101) | cost, Appendix C.4 (pp. 158–162)

Stochastic unit parsers

One string can have several segmentations.

乒乓球拍卖完了

乒乓球 | 拍卖 | 完了

ping-pong ball · auction · finished
“The ping-pong ball auction is over.”

乒乓球拍 | 卖 | 完了

ping-pong paddle · sell · finished
“The ping-pong paddles have been sold.”

In general a unit parser is a distribution over segmentations. The thesis treats the deterministic case.

The extension is a weighted transducer: one source string, several weighted outputs.

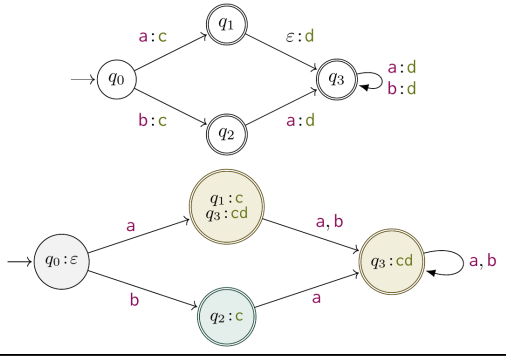
Thesis Example 6.1 (p. 88) | deterministic assumption (p. 89) | Section 6.1.2 (p. 91) | future work, Section 7.4 (p. 108)

The decomposition algorithm

```
def decompose(y): # memoize
  N ← |y|
  if N = 0:
    q ← QUEUE({ε})
  else:
    (Q′, R′) ← decompose(y_{<N})
    q ← QUEUE(Q′ ∪ R′)
  (Q, R) ← (∅, ∅)
  while |q| > 0:
    q′ ← ∅
    for x ∈ q:
      if is_cylinder(x, y):
        Q.add(x)
        continue
      if is_member(x, y):
        R.add(x)
      for x′ ∈ 𝒳:
        if is_live(xx′, y):
          q′.add(xx′)
    q ← prune(q′)
  return (Q, R)
```


Thesis: Algorithm 4.2 (p. 36). Snæbjarnarson, Kiegeland et al. (ICLR 2026), Fig. 2

Frontier example and lazy determinization



x	Frontier	is_live	is_cylinder	is_member	Class
ε	$\{(q_0, \varepsilon)\}$	✓	✗	✗	extend
a	$\{(q_1, \mathsf{c}), (q_3, \mathsf{cd})\}$	✓	✓	—	$\mathcal{Q}(\mathsf{c})$
b	$\{(q_2, \mathsf{c})\}$	✓	✗ (q_2 dead on b)	✓	$\mathcal{R}(\mathsf{c})$

Thesis: Section 4.3.1 (pp. 42–48). Snæbjarnarson, Kiegeland et al. (ICLR 2026), Fig. 8

Beam summing for transduced language models

```
def beam_summing(y, ψ, prune):
  N ← |y|
  if N = 0:
    (Q′, R′), Ẑ ← ({(ε, 1)}, ∅), [ ]
  else:
    (Q′, R′), Ẑ ←
      beam_summing(y_{<N}, ψ, prune)
  P ← QUEUE(Q′)
  (Q, R), A ← (∅, ∅), 0
  for (x, w) ∈ R′:
    if is_member(x, y):
      R.add((x, w))
      A += w
  while |P| > 0:
    P′ ← ∅
    for x, w ∈ P:
      if is_cylinder(x, y):
        Q.add((x, w))
        A += w
        continue
      if is_member(x, y):
        R.add((x, w · p⃗_K(EOS | x)))
        A += w · p⃗_K(EOS | x)
      for x′ ∈ X with ψ(xx′) > 0:
        P′.add((xx′, w · p⃗_K(x′ | x)))
    P ← prune(P′, A)
  Ẑ.append(A)
  return (Q, R), Ẑ
```


Thesis: Algorithm 5.1 (p. 65). Snæbjarnarson et al. (ICLR 2027 submission), Fig. 1

Sampling without replacement

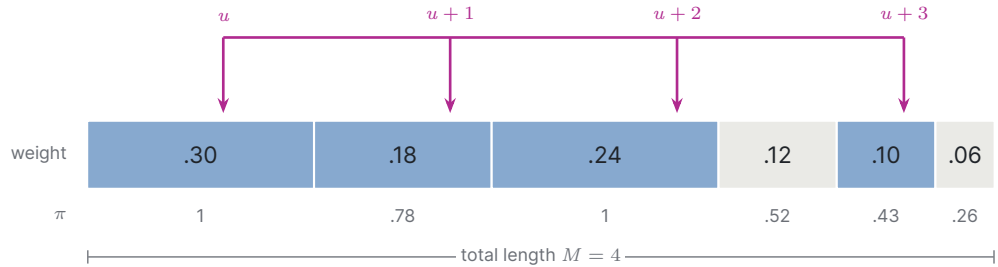
29 **def** swor$_{M,\psi}(\mathscr{P}', A)$:
30 $\quad \mathcal{W} \leftarrow (w^{(i)} \cdot \psi(\boldsymbol{x}^{(i)}))_{i=1}^{|\mathscr{P}'|}$
31 $\quad I_+ \leftarrow \{i : W^{(i)} > 0\}$
32 $\quad$ **if** $|I_+| \leq M$:
33 $\quad\quad$ **return** $\{(\boldsymbol{x}^{(i)}, w^{(i)}) : i \in I_+\}$
34 $\quad \pi \leftarrow \mathsf{pps}(\mathcal{W}, M)$
35 $\quad \mathcal{S} \sim \mathsf{systematic}(\pi,\ M)$
36 $\quad$ **return** $\{(\boldsymbol{x}^{(i)}, \frac{w^{(i)}}{\pi_i}) : i \in \mathcal{S}\}$

37 **def** beam_swor$(\boldsymbol{y}, M, \psi)$:
38 $\quad$ **return** beam_summing$(\boldsymbol{y}, \psi,$ swor$_{M,\psi})$
39
40 **def** beam_swor_ad$(\boldsymbol{y}, M, \psi)$:
41 $\quad$ **return**
42 $\quad\quad$ beam_summing$(\boldsymbol{y}, \psi,$ swor_adaptive$_{M,\psi,\rho})$

43 **def** swor_adaptive$_{M,\psi,\rho}(\mathscr{P}', A)$:
44 $\quad \mathcal{W}^{(i)} \leftarrow w^{(i)} \cdot \psi(\boldsymbol{x}^{(i)})$
45 $\quad W \leftarrow \sum_i \mathcal{W}^{(i)}$
46 $\quad I_+ \leftarrow \{i : W^{(i)} > 0\}$
47 $\quad$ **if** $W = 0$:
48 $\quad\quad$ **return** $\emptyset$
49 $\quad \mu \leftarrow \sum_i \min(1, \frac{\mathcal{W}^{(i)} M}{\rho\,(A+W)})$
50 $\quad m \leftarrow \min(M, \lfloor \mu + \frac{1}{2} \rfloor)$
51 $\quad$ **if** $m = 0$:
52 $\quad\quad B \sim \mathrm{Bernoulli}(\mu)$
53 $\quad\quad$ **if** $B = 0$: **return** $\emptyset$
54 $\quad\quad m \leftarrow 1, \quad w^{(i)} \leftarrow w^{(i)}/\mu \quad (\forall i)$
55 $\quad$ **if** $|I_+| \leq m$:
56 $\quad\quad$ **return** $\{(\boldsymbol{x}^{(i)}, w^{(i)}) : i \in I_+\}$
57 $\quad \pi \leftarrow \mathsf{pps}(\mathcal{W}, m)$
58 $\quad \mathcal{S} \sim \mathsf{systematic}(\pi,\ m)$
59 $\quad$ **return** $\{(\boldsymbol{x}^{(i)}, \frac{w^{(i)}}{\pi_i}) : i \in \mathcal{S}\}$

Thesis: Algorithm 5.2 (p. 69). Snæbjarnarson et al. (ICLR 2027 submission), Fig. 2

Systematic sampling: one draw for the whole pool



Each particle gets a segment as long as its inclusion probability. One uniform number u places a comb with teeth one unit apart.

A particle is kept when a tooth lands in its segment: exactly M are kept, none twice, and a segment of length 1 always catches a tooth.

Thesis: Algorithm 5.2 (p. 69). Madow (1949)

The smc_rb algorithm

```
def smc_rb(y, M, ψ, prune):
  N ← |y|
  if N = 0:
    (Q′, R′), Ẑ ← ({(ε, 1/M)}_{m=1}^{M}, ∅), [ ]
  else:
    (Q′, R′), Ẑ ← smc_rb(y_{<N}, M, ψ, prune)
  P ← QUEUE(Q′)
  (Q, R), A ← (∅, ∅), 0
  for (x, w) ∈ R′:
    if is_member(x, y):
      R.add((x, w))
      A += w
  while |P| > 0:
    P′ ← ∅
    for x, w ∈ P:
      if is_cylinder(x, y):
        Q.add((x, w))
        A += w
        continue
      if is_member(x, y):
        R.add((x, w · p⃗_X(EOS | x)))
        A += w · p⃗_X(EOS | x)
      λ ← Σ_{x′∈X} p⃗_X(x′ | x) · ψ(xx′)
      if λ = 0:
        continue
      sample x′ ~ p⃗_X(· | x) ψ(x·)/λ
      P′.add((xx′, w·λ/ψ(xx′)))
    P ← prune(P′, A)
  Ẑ.append(A)
  return (Q, R), Ẑ
```

```
def resample_{ψ,η}(P′, A):
  W^(i) ← w^(i) · ψ(x^(i))
  V ← Σ_i W^(i)
  N ← |P′|
  if V² ≥ η N Σ_i (W^(i))²:
    return P′
  j_1, ..., j_N ~ i.i.d. Categorical(W^(i)/V)
  return {(x^(j_n), w^(j_n) V / N W^(j_n))}_{n=1}^{N}
```


Thesis: Algorithm B.2 (p. 138). Snæbjarnarson et al. (ICLR 2027 submission), Fig. 10

The cost is linear in the size of the machine



states $S = 4$ arcs $E = 4$

$$O\left(E + S\,|\Sigma_X|\right)$$

one pass over every arc, and over every state for each source symbol (after ε-removal)