aivancity
SCHOOL FOR
TECHNOLOGY, BUSINESS & SOCIETY
PARIS-CACHAN

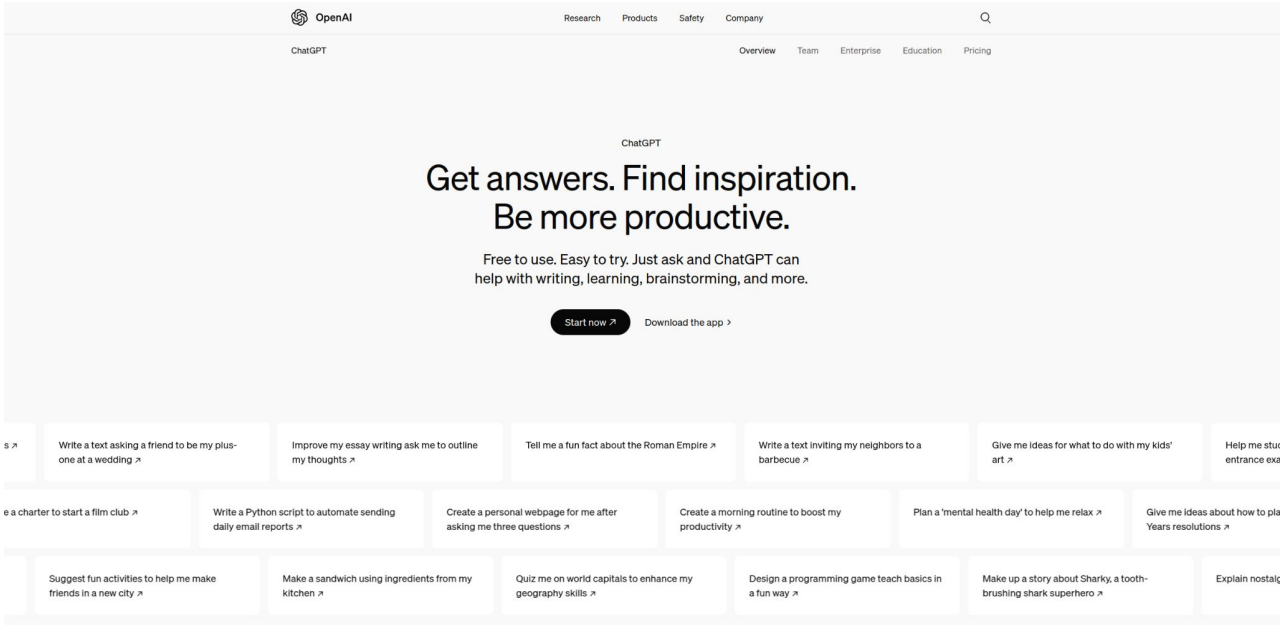
23/10/2025

Natural Language Processing (NLP)

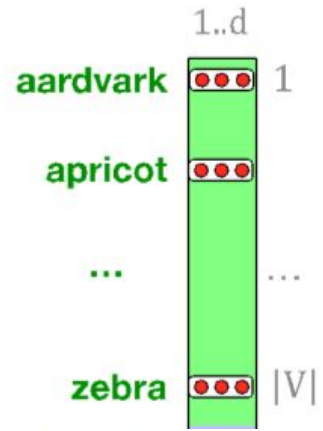
N-gram and Neural Language Models: from Shannon to ChatGPT

Part 1

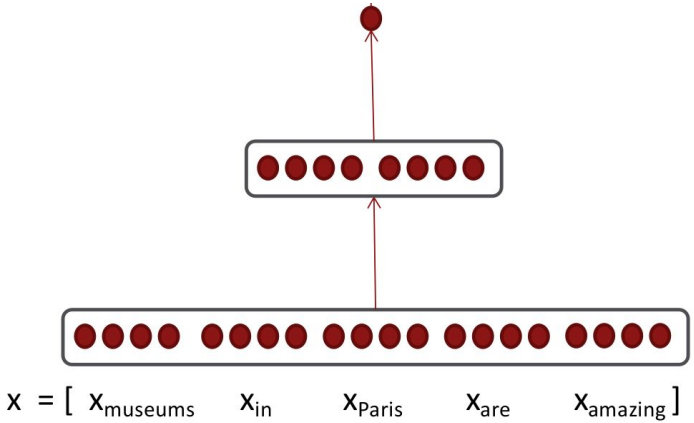
Chatbots like ChatGPT rely on LLMs



What can we do so far?



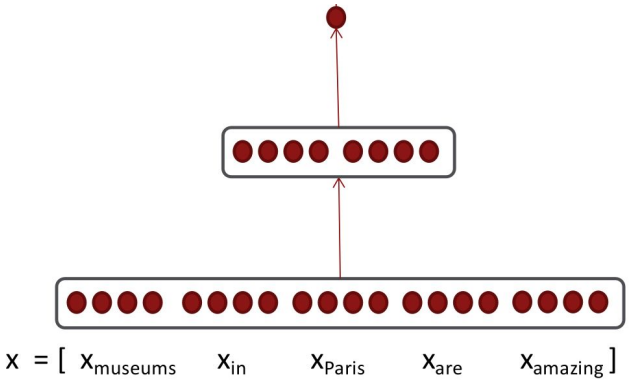
- Embedding matrix: a vector for each word
- Fine for **classification** (e.g. Named Entity Recognition)



Part 1

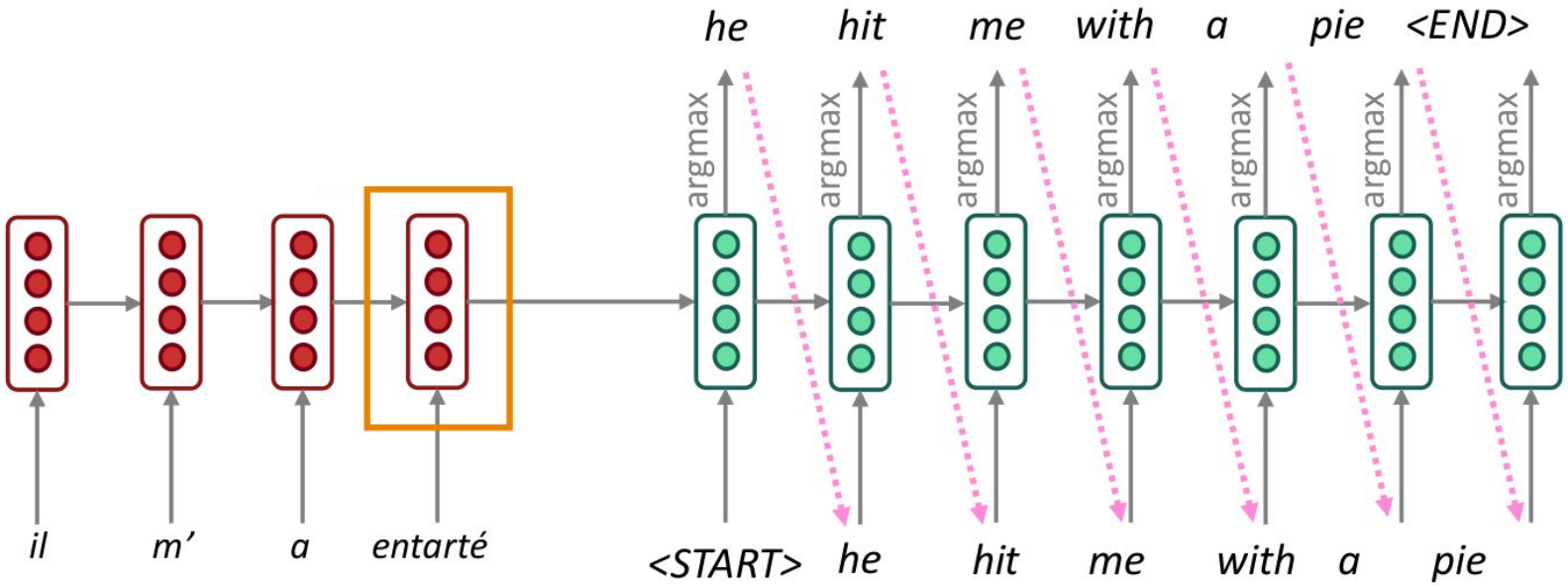
Continuous bag of words (CBOW)

"bag of words" because does not model word order, puts all words in the same "bag"



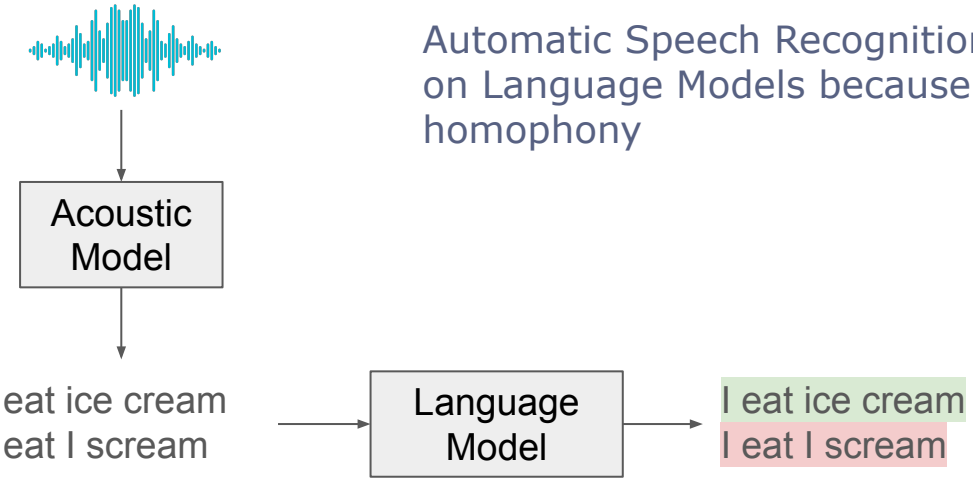
Part 1

Sequence-to-Sequence (Translation)



Part 1

Why Language Modeling?



Automatic Speech Recognition (ASR) relies on Language Models because of homophony

Part 1

Language Compositionality

- Assume a (finite) vocabulary of words

$$\mathcal{V} = \{\text{I}, \text{eat}, \text{ice}, \text{cream}, \text{scream} \ldots\}$$

- We can construct an (infinite) set of strings

$$\mathcal{V}^{\dagger} = \{\text{I}, \text{eat}, \text{I eat}, \text{eat I}, \text{I cream}, \text{ice cream}, \text{ice scream} \ldots\}$$

Interpret "I eat ice cream"

Interpret "I eat pasta"

→ know meaning of "ice cream"/"pasta"

→ know relation between the subject "I", the verb "to eat" and the object "ice cream"/"pasta"

Language Modeling

- Assume a (finite) vocabulary of words

  $\mathcal{V} = \{\text{I}, \text{cream}, \text{eat}, \text{ice}, \text{scream}, \text{pasta} \ldots\}$

- We can construct an (infinite) set of strings

$$\mathcal{V}^\dagger = \{\text{I}, \text{eat}, \text{I eat}, \text{eat I}, \text{I cream}, \text{ice cream}, \text{ice scream} \ldots\}$$

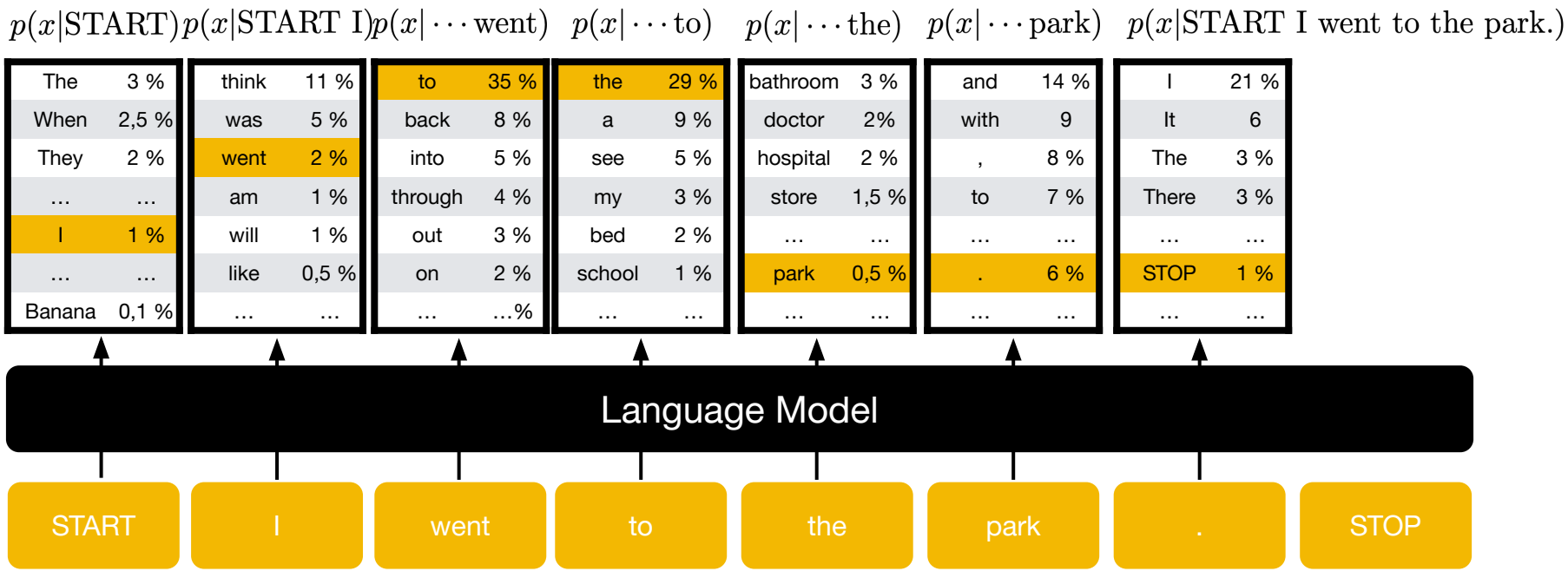
Input: training data $\boldsymbol{x} = \langle x_1, \ldots, x_N \rangle$ in $\mathcal{V}^\dagger$

Output: a function $p : \mathcal{V}^\dagger \to \mathbb{R} \quad \forall \boldsymbol{x} \in \mathcal{V}^\dagger, p(\boldsymbol{x}) \geq 0$

$$\sum_{\boldsymbol{x} \in \mathcal{V}^\dagger} p(\boldsymbol{x}) = 1$$

Part 1

Learning from sequences: the Chain Rule



Learning from sequences: the Chain Rule

$$p(\boldsymbol{X} = \boldsymbol{x}) = \left(\begin{array}{l} p(X_1 = x_1) \\ \cdot\ p(X_2 = x_2 \mid X_1 = x_1) \\ \cdot\ p(X_3 = x_3 \mid \boldsymbol{X}_{1:2} = \boldsymbol{x}_{1:2}) \\ \vdots \\ \cdot\ p(X_N = \bigcirc \mid \boldsymbol{X}_{1:N-1} = \boldsymbol{x}_{1:N-1}) \end{array} \right)$$

$$= \prod_{i=1}^{N} p(X_i = x_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1})$$

n-gram Models (Markov assumption)

Unigram (n=1): No history! All words are independent!

$$p(\boldsymbol{X} = \boldsymbol{x}) = \prod_{i=1}^{N} p(X_i = x_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1}) \overset{\text{assumption}}{=} \prod_{i=1}^{N} p(X_i = x_i; \boldsymbol{\theta})$$

$$p(\mathbf{X} = \mathrm{I\ eat\ I\ scream}) = p(X_1 = \mathrm{I}) \cdot p(X_2 = \mathrm{eat}) \cdot p(X_3 = \mathrm{I}) \cdot p(X_4 = \mathrm{scream})$$

Part 1

Remember Bag of Words / Naive Bayes

$$P(x_1, x_2, \ldots, x_n \mid c)$$

Bag of Words/Naive Bayes assumption: Assume position doesn't matter

Conditional Independence: Assume the feature probabilities $P(x_i \mid c_j)$ are independent given the class c.

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

n-gram Models (Markov assumption)

$$p(\boldsymbol{X} = \boldsymbol{x}) = \prod_{i=1}^{N} p(X_i = x_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1})$$

$$\overset{\text{assumption}}{=} \prod_{i=1}^{N} p(X_i = x_i \mid X_{i-\mathsf{n}+1:i-1} = \boldsymbol{x}_{i-\mathsf{n}+1:i-1}; \boldsymbol{\theta})$$

Bigram (n=2):

$$p(\mathbf{X} = \text{I eat I scream}) = p(X_1 = \text{I}|x_0) \cdot p(X_2 = \text{eat}|\text{I}) \cdot p(X_3 = \text{I}|\text{eat}) \cdot p(X_4 = \text{scream}|\text{I})$$

Trigram (n=3):

$$p(\mathbf{X} = \text{I eat I scream}) = p(X_1 = \text{I}|x_0\, x_0) \cdot p(X_2 = \text{eat}|x_0\, \text{I}) \cdot p(X_3 = \text{I}|\text{I eat}) \cdot p(X_4 = \text{scream}|\text{eat I})$$

Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

An example

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$$

Part 1

More examples: Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- tell me about chez panisse
- i’m looking for a good place to eat breakfast
- when is caffe venezia open during the day

Part 1

Raw bigram counts → sparsity

- Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Sparsity in n-grams

If n is too small, your model can't learn very much about language.

As n gets larger:

- The number of parameters grows with $O(V^{\mathrm{n}})$.
  - What's a parameter?
- Most n-grams will never be observed, so you'll have lots of zero probability n-grams. This is an example of **data sparsity**.
- Your model depends increasingly on the training data; you need (lots) more data to learn to generalize well.

Part 1

Bias-variance tradeoff in n-grams

- n is small → underfit training data, high bias, low variance

  unigram "the" is highly probable, regardless of your corpus
- n is large → overfit training data, low bias, high variance

  a 7-gram like "my name is Paul I'm a postdoc" has 0 hits on Google

  but "my name is Paul I'm a *student*" has several → somewhat probable

  but "my name is *Albertine* I'm a student" has 0 hits on Google

Perplexity

The Shannon Game:

- How well can we predict the next word?

  When I eat pizza, I wipe off the ___

  Many children are allergic to ___

  I saw a ___

- Unigrams are terrible at this game. (Why?)

grease	0.5
sauce	0.4
dust	0.05
…	
mice	0.001
…	
the	1E-100

Claude Shannon

<u>How good are we doing?</u>

- Want to assign *high probability* to observed words

Part 1

Perplexity

Given a test dataset $\bar{\boldsymbol{x}}$ (of $\bar{N}$ words), we arrive at the standard intrinsic evaluation in three steps:

1. Probability of the test data: $p(\bar{\boldsymbol{x}}; \boldsymbol{\theta})$
2. That value will be tiny, because $\mathcal{V}^\dagger$ is infinitely large, and p will decrease exponentially in the length of $\bar{\boldsymbol{x}}$. So we transform it:

$$\text{Perplexity}(\bar{\boldsymbol{x}}; p(\cdot; \boldsymbol{\theta})) = \sqrt[\bar{N}]{\frac{1}{p(\bar{\boldsymbol{x}}; \boldsymbol{\theta})}} = 2^{\left(\frac{1}{\bar{N}} \times -\log_2 p(\bar{\boldsymbol{x}}; \boldsymbol{\theta})\right)}$$

Cross-Entropy

Special cases:

- If the model were to put *all* of its probability on $\bar{\boldsymbol{x}}$, perplexity would be 1 (minimal possible value).
- If the model assigns zero probability to $\bar{\boldsymbol{x}}$, perplexity is $+\infty$. So it's important to make sure that p assigns strictly positive probability to *every* sequence of words.

You can interpret perplexity as "effective size of the vocabulary."

Perplexity

Consider a unigram model that is completely agnostic; it assigns $\theta_v = \frac{1}{V}$ for all $v \in \mathcal{V}$.

What will its perplexity be? Hint: as long as the test data is restricted to words in $\mathcal{V}$, the test data doesn't matter!

$$\text{Perplexity}(\bar{\boldsymbol{x}}; p(\cdot; \boldsymbol{\theta})) = \sqrt[\bar{N}]{\frac{1}{p(\bar{\boldsymbol{x}}; \boldsymbol{\theta})}} = \sqrt[\bar{N}]{\frac{1}{\left(\frac{1}{V}\right)^{\bar{N}}}} = V$$

Perplexity

- How "hard" is the task of recognizing digits '0,1,2,..,9' uniformly at random?

$d \sim \text{Uniform}(0, 9)$

$$\text{PP}(d_1, \ldots, d_N) = p(d_1, \ldots, d_N)^{-\frac{1}{N}} = \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} = \frac{1}{10}^{-1} = 10$$

  - Perplexity: 10
  - Using entropy (replacing the estimated distribution with the known true dist.):

$$H(D, D) = -\sum_{d \in D} p(d) \log p(d) = -\sum_{i=0}^{9} p(i) \log p(i) = -\sum_{i=0}^{9} \frac{1}{10} \log\left(\frac{1}{10}\right) = -\log\left(\frac{1}{10}\right)$$

$$PP(D) = e^{H(D,D)} = e^{-\log\left(\frac{1}{10}\right)} = \left(e^{\log\left(\frac{1}{10}\right)}\right)^{-1} = \left(\frac{1}{10}\right)^{-1} = 10$$

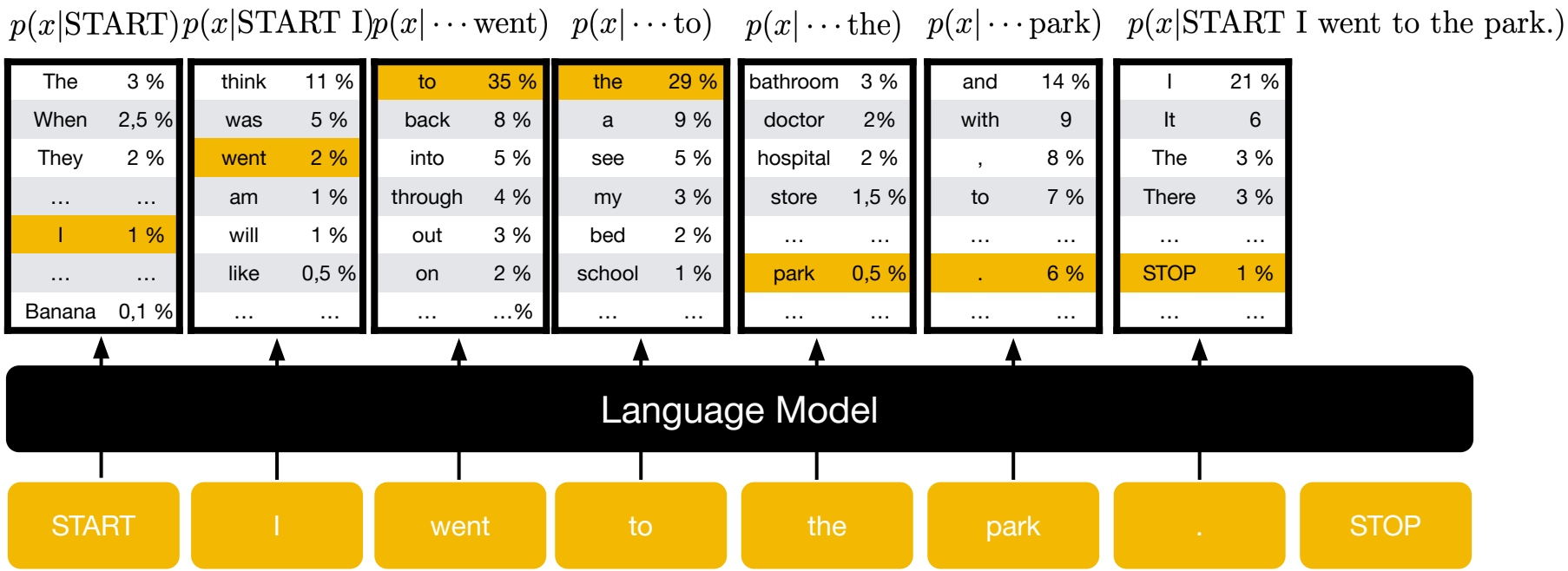
Same result!

Part 2

Break for questions and "appel"

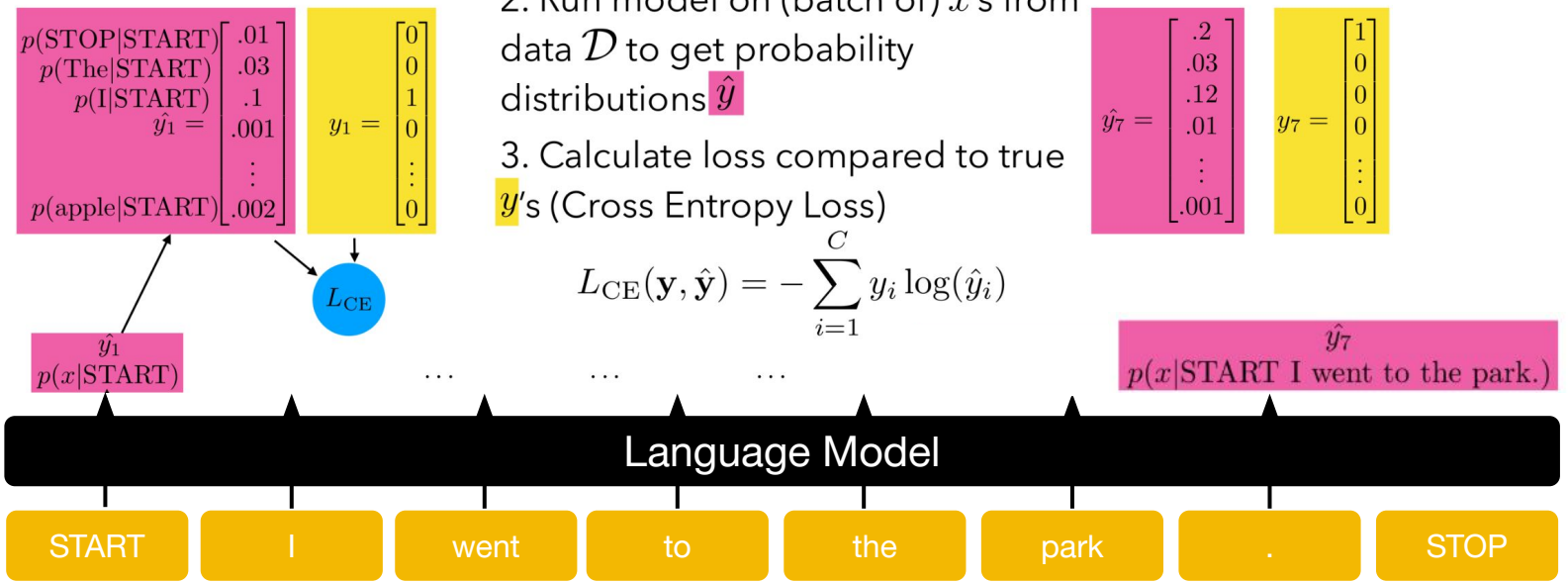
Part 2

Language Modeling (no Markov assumption)



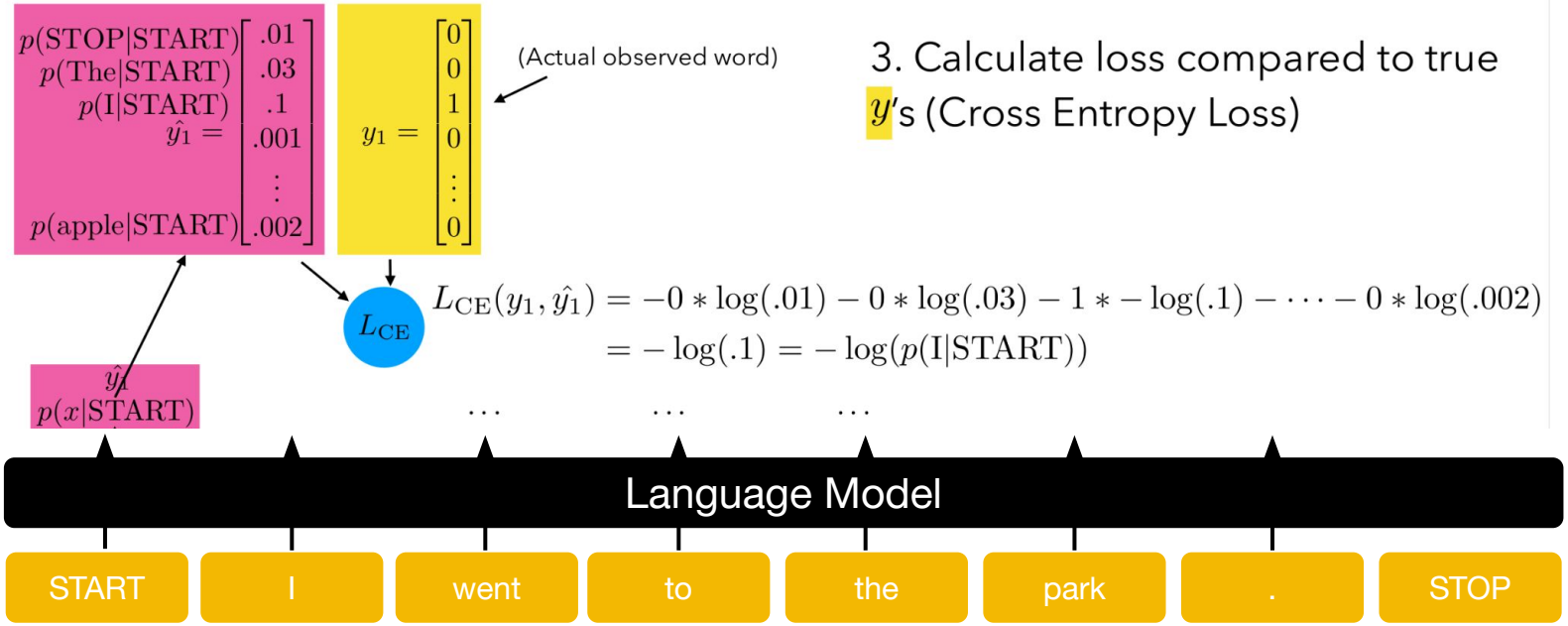
Part 2

Generation as a Sequence of Classifications



Cross-Entropy

$$L_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$



Part 2

Pretraining (for Transfer Learning)

- **Self-supervision**: Language Models do not need annotated data to be pretrained, only raw text, and there are plenty
- Deep Learning thrives on data (the more the better, more complex models without overfitting)
- Models can then be **fine-tuned** on a downstream task (e.g. Named Entity Recognition).
- Much better than training from scratch because annotated data is rare (almost nonexistent except in English) because it's expensive

LLMs learn about syntax and semantics

- Syntax :

```
I eat
I ate
I am
I eats
eat I
```

- Semantic :

```
I eat bread
I ate pasta
I ate sand
I am a student
I am a fork
I ate pasta with my fork
I ate pasta with my keyboard
```

Part 2

LLMs learn about much more

- Entities :

```
My favorite city is Paris
My favorite city is France
I am visiting Paris, the capital of France
I am visiting Paris, the capital of Italy
```

- Reasoning :

```
All men are mortal. Socrates is a man.
Therefore, Socrates is mortal
```

Part 2

Autoregressive generation

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: **After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.**

GPT-3 (2020)

Part 2

Every NLP task as Autoregressive generation

- Classification: "I like this movie"

  → `"I like this movie, it was {good/bad}"`

- Question Answering: "When was Dante born?"

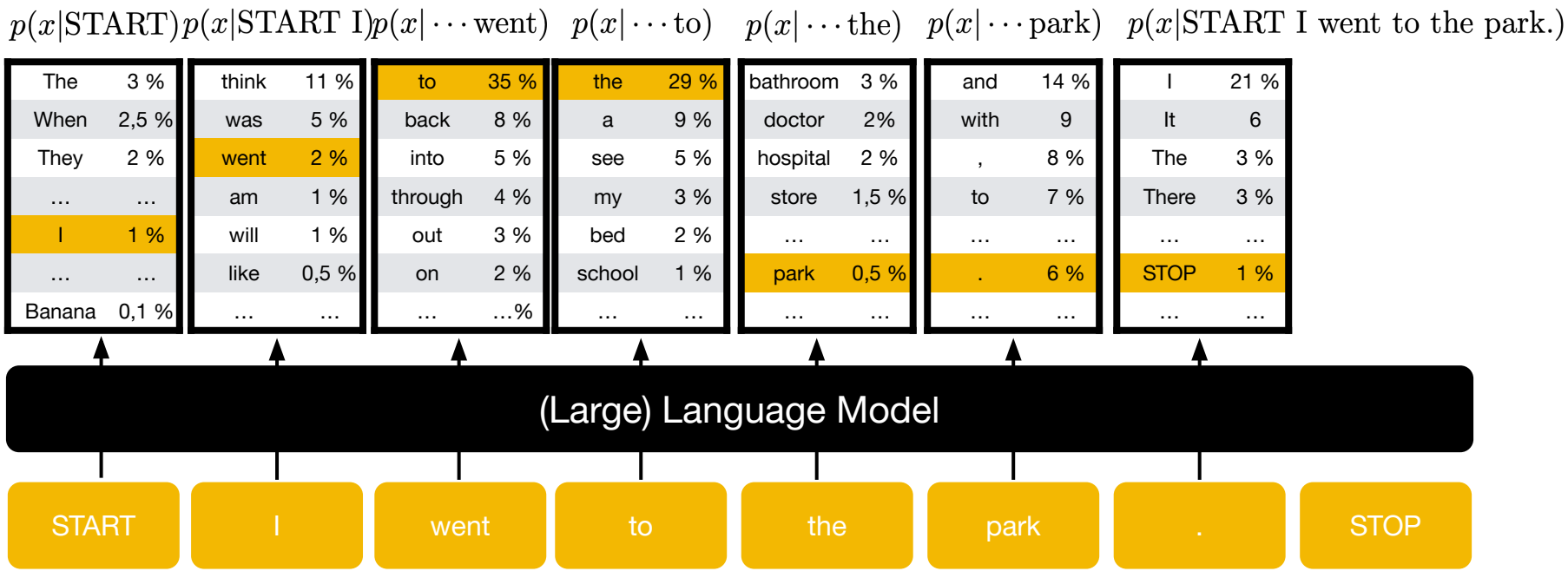
  → `"Dante was born in ____"`

- Translation: "I like pasta"

  → `"The translation of 'I like pasta' in French is ____"`

Part 3

Decoding from a Language Model



Part 3

Decoding from a Language Model

- At each time step t, our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$:

$$S = \underbrace{f(\{y_{<t}\}; \theta)}_{f(\cdot\,;\theta) \text{ is your model}}$$

- Then, we compute a probability distribution P over $w \in V$ using these scores:

$$P(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Our decoding algorithm defines a function to select a token from this distribution:

$$\hat{y}_t = \underbrace{g(P(y_t \mid \{y_{<t}\}))}_{g(\cdot) \text{ is your decoding algorithm}}$$

Part 3

Maximum A Posteriori (MAP) Decoding: Getting the most likely output

- **Obvious method: Greedy Decoding**
  - Selects the highest probability token according to $P(y_t \mid y_{<t})$

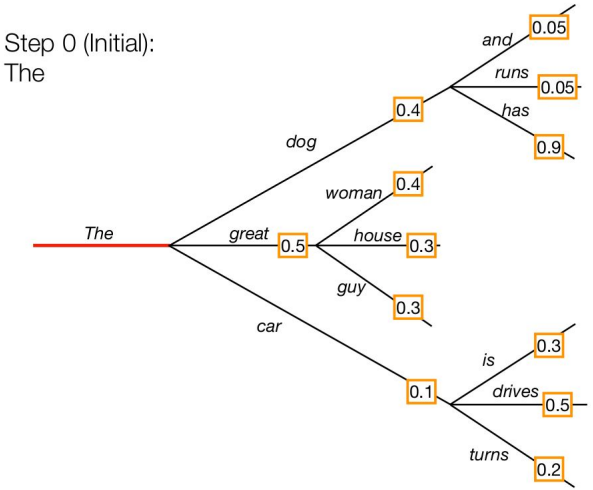
$$\hat{y}_t = \textbf{argmax}_{w \in V} P(y_t = w \mid y_{<t})$$

- **Beam Search**
  - Also aims to find the string with the highest probability, but with a wider exploration of candidates.

Part 3

Greedy Decoding vs. Beam Search

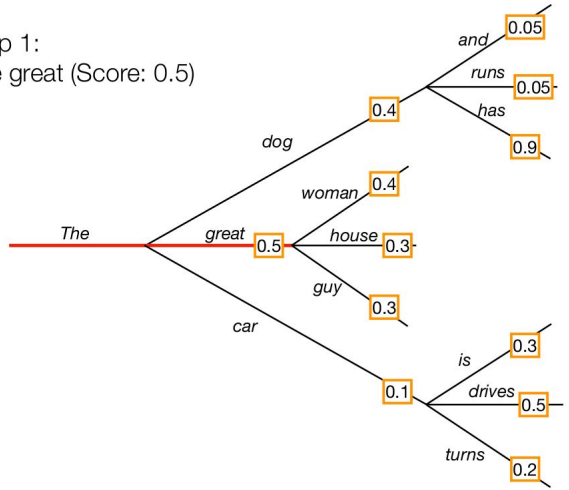
- **Greedy Decoding**
  - Choose the "currently best" token at each time step



Part 3

Greedy Decoding vs. Beam Search

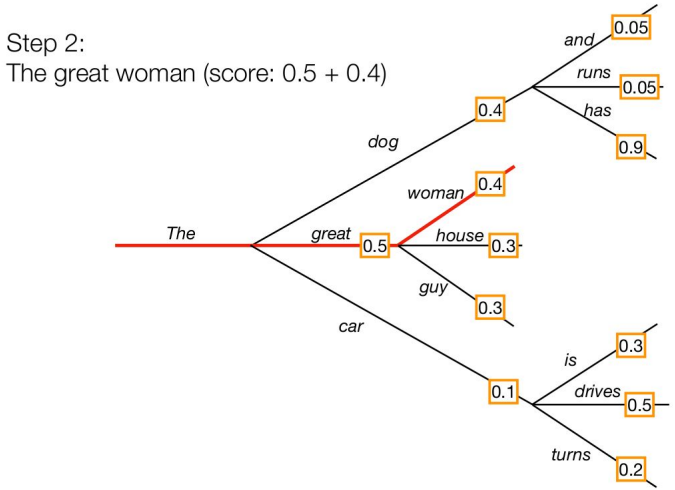
- **Greedy Decoding**
  - Choose the "currently best" token at each time step



Part 3

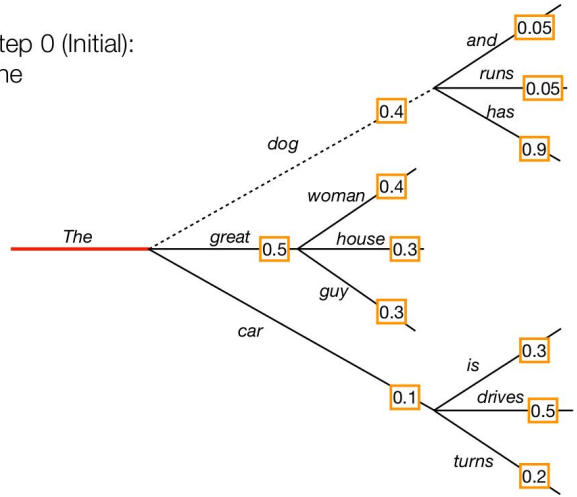
Greedy Decoding vs. Beam Search

- **Greedy Decoding**
  - Choose the "currently best" token at each time step



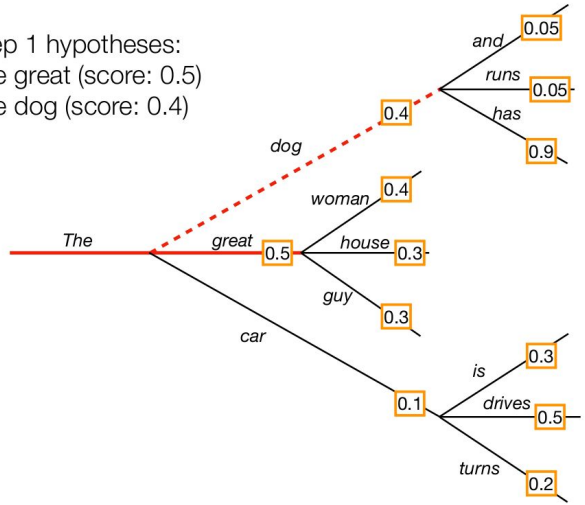
Greedy Decoding vs. Beam Search

- **Beam Search (in this example, *beam_width* = 2)**
  - At each step, retain 2 hypotheses with the highest probability



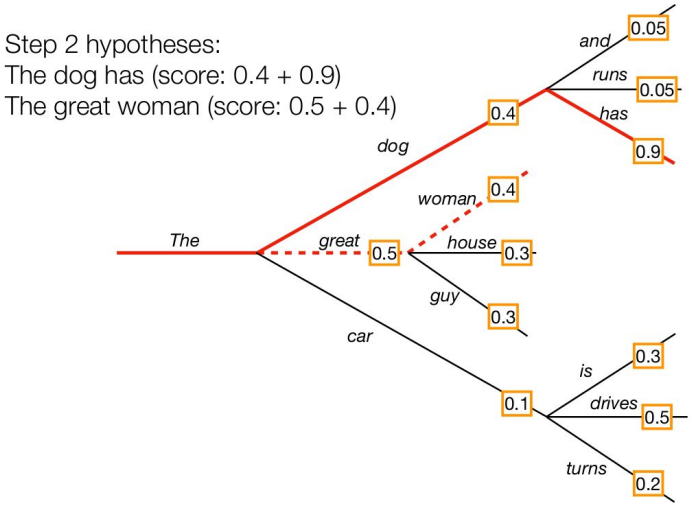
Greedy Decoding vs. Beam Search

- **Beam Search (in this example, *beam_width* = 2)**
  - At each step, retain 2 hypotheses with the highest probability



Greedy Decoding vs. Beam Search

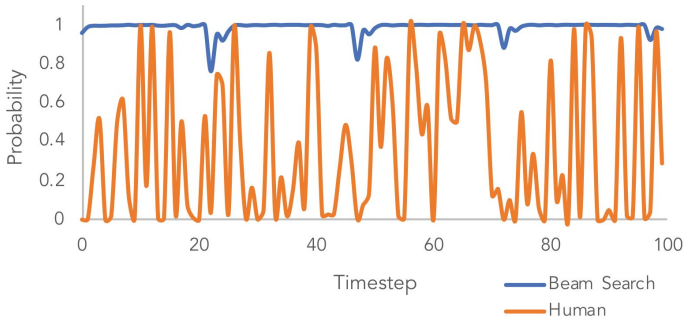
- **Beam Search (in this example, *beam_width* = 2)** A type of Best-First Search
  - At each step, retain 2 hypotheses with the highest probability



Part 3

Maximum A Posteriori (MAP) Decoding: Getting the most likely output

- Great for constrained tasks, e.g. summarization and translation
- Bad for open-ended generation, e.g. chatbot



Greedy methods fail to capture the variance of human text distribution.

Part 3

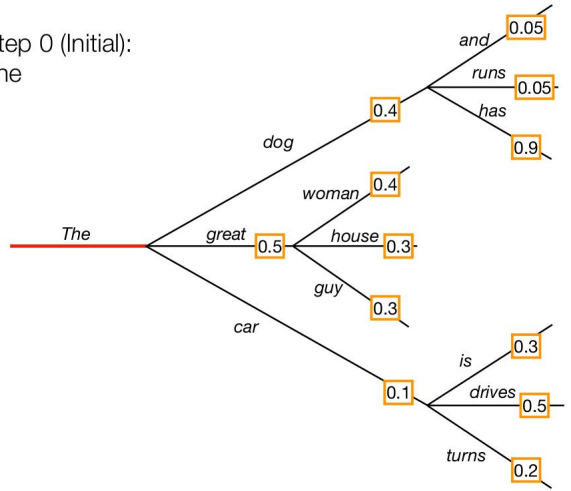
Adding randomness (aiming for humanness)

- Sample a token from the token distribution at each step!

$$\hat{y}_t \sim P(y_t = w \mid \{y\}_{<t})$$

- It's inherently *random* so you can sample any token.

```
the dog has      (0.40*0.90=0.36)
the dog has      (0.40*0.90=0.36)
the great woman  (0.50*0.40=0.20)
the great house  (0.50*0.30=0.15)
the great guy    (0.50*0.30=0.15)
the dog runs     (0.40*0.05=0.02)
```



Part 3

Scaling randomness: Softmax temperature

- Recall: At time step t, model computes a distribution P_t by applying softmax to a vector of scores $S \in \mathbb{R}^{|V|}$

$$P_t(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Here, you can apply **temperature hyperparameter** τ to the softmax to rebalance P_t:

$$P_t(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

- Raise the temperature $\tau > 1$: P_t becomes more uniform
  - More diverse output (probability is spread across vocabulary)
- Lower the temperature $\tau < 1$: P_t becomes more spiky
  - Less diverse output (probability concentrated to the top tokens)

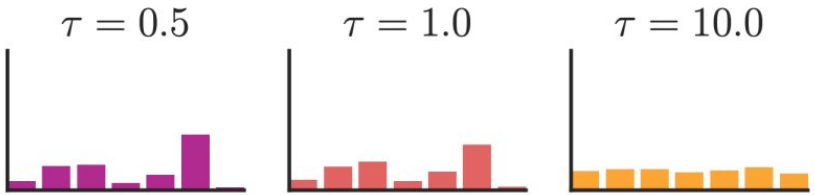
Part 3

Scaling randomness: Softmax temperature

- You can apply **temperature hyperparameter** τ to the softmax to rebalance P_t:

$$P_t(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

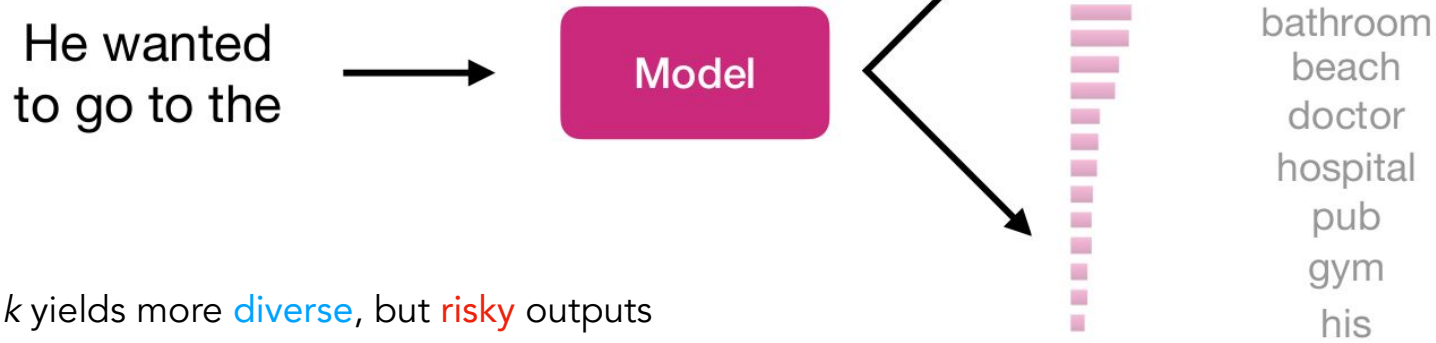
- Raise the temperature $\tau > 1$: P_t becomes more uniform
  - More diverse output (probability is spread across vocabulary)
- Lower the temperature $\tau < 1$: P_t becomes more spiky
  - Less diverse output (probability concentrated to the top tokens)



Part 3

With a threshold so it's not too random

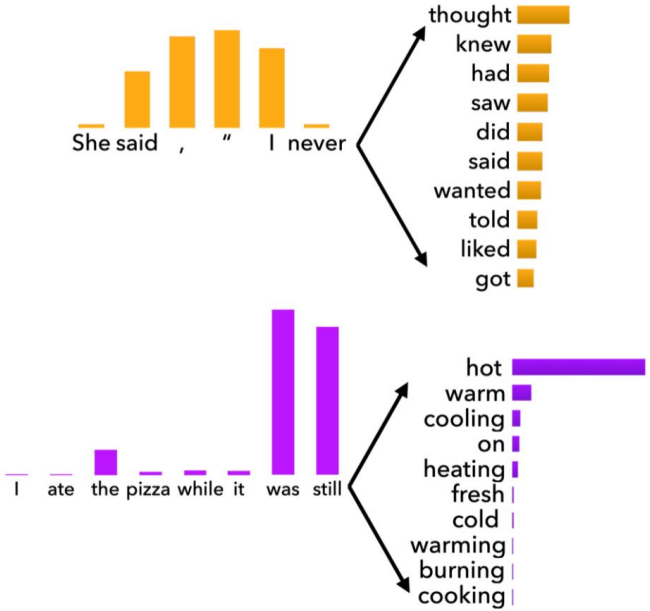
- Solution: Top-*k* sampling *(Fan et al., 2018)*
  - Only sample from the top *k* tokens in the probability distribution.
  - Common values for *k* = 10, 20, 50 (*but it's up to you!*)



  - Increasing *k* yields more diverse, but risky outputs
  - Decreasing *k* yields more safe but generic outputs

Part 3

How to set the threshold?



For *flat* distribution,
Top-*k* Sampling may cut off too **quickly**!

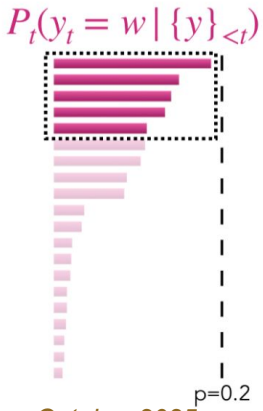
For *peaked* distribution,
Top-*k* Sampling may also cut off too **slowly**!

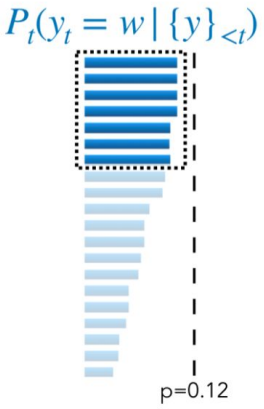
Part 3

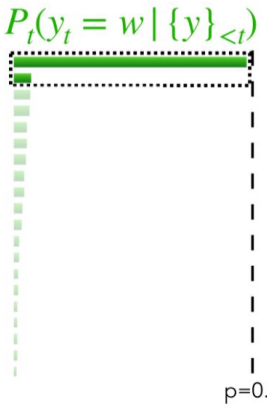
Vary the threshold! Nucleus Sampling (top-p)

- Solution: Top-p sampling *(Holtzman et al., 2020)*
  - Sample from all tokens in the top p cumulative probability mass (i.e., where mass is concentrated)
  - Varies k according to the uniformity of P_t

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p$$







Part 3

Beam Search vs. Nucleus Sampling

WebText

An unprecedented number of mostly young whales have become stranded on the West Australian coast since 2008.

Beam Search, b=16

The number of stranded whales has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year. The number of whales stranded on the West Australian coast has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year.

Nucleus, p=0.95

There has been an unprecedented number of calves caught in the nets of whaling stations that operate in WA. Pilot whales continue to migrate to feeding grounds to feed their calves. They are now vulnerable due to the decline of wild populations; they are restricted to one breeding site each year. Image copyright Yoon Bo Kim But, with sharp decline in wild populations the size of the Petrels are shrinking and dwindling population means there will only be room for a few new fowl.

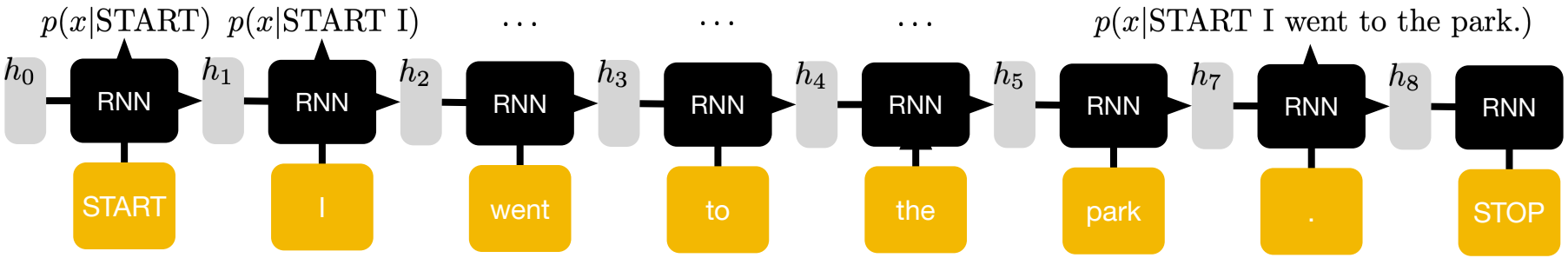
Part 3

Summarizing

- Language models learn which strings are likely or unlikely, useful for:
  - Automatic Speech Recognition (because of homophony)
  - Pretraining neural networks without any labeled data (self-supervision)
  - Any NLP task can be set as text-to-text/cloze test
- n-gram have Markov assumption → count occurrences
- Neural models use stochastic gradient descent
- Decoding the most likely string is good for factual tasks (Question Answering, Translation), but not for chit-chat

Part 1

Next class: Recurrent Neural Networks!



Part 0

Acknowledgements

This class directly builds upon:

- **Jurafsky, D., & Martin, J. H.** (2024). Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models (3rd éd.).
- **Eisenstein, J.** (2019). Natural Language Processing. 587.
- **Yejin Choi**. (Winter 2024). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Noah Smith**. (Winter 2023). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Benoît Sagot**. (2023-2024). *Apprendre les langues aux machines* (Collège de France)
- **Chris Manning**. (Spring 2024). Stanford CS224N: Natural Language Processing with Deep Learning
- Classes where I was/am Teacher Assistant:
  - **Christopher Kermorvant**. Machine Learning for Natural Language Processing (ENSAE)
  - **François Landes** and **Kim Gerdes**. Introduction to Machine Learning and NLP (Paris-Saclay)

Also inspired by:

- My PhD thesis: *Répondre aux questions visuelles à propos d'entités nommées* (2023)
- **Noah Smith** (2023): Introduction to Sequence Models (LxMLS)
- **Kyunghyun Cho**: Transformers and Large Pretrained Models (LxMLS 2023), Neural Machine Translation (ALPS 2021)
- My former PhD advisors **Olivier Ferret** and **Camille Guinaudeau** and postdoc advisor **François Yvon**
- My former colleagues at LISN

aivancity

PARIS-CACHAN

advancing education
in artificial intelligence

Part 1

LSTM with Attention: formally

We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$

On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$

We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Sequence-to-Sequence (Translation)