CS 784: Computational Linguistics
Lecture 10: Language Models

Freda Shi

School of Computer Science, University of Waterloo
fhs@uwaterloo.ca

February 24, 2025

Recap: Distributional Semantics

You shall know a word by the company it keeps.

– J. R. Firth, 1957

- A bottle of **tezgüino** is on the table.
- Everybody likes **tezgüino**.
- Don't have too much **tezgüino** before you drive.
- **Tezgüino** is made out of corn.

- A bottle of ____ is on the table.
- Everybody likes ____.
- Don't have too much ____ before you drive.
- ____ is made out of corn.

This is **language modeling**.

Language Models

Language model: a probability distribution over strings in a language.

$$P(\mathbf{x}) = P(x_1, x_2, \ldots, x_T)$$

$$P(\textit{The cat is cute.}) = 0.00000004$$
$$P(\textit{I am hungry.}) = 0.0000001$$
$$P(\textit{Dog the asd@sdf 1124?!!?}) \approx 0$$

Language modeling: the task of estimating this string distribution from data.

- Define a statistical model $P_{\mathbf{\Theta}}(\mathbf{x})$ ($\mathbf{x}$: string).
- Estimate the parameters $\mathbf{\Theta}$ from data (by maximizing likelihood).

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} \prod_{i=1}^{N} P_{\mathbf{\Theta}}(\mathbf{x}_i) = \arg\max_{\mathbf{\Theta}} \sum_{i=1}^{N} \log P_{\mathbf{\Theta}}(\mathbf{x}_i)$$

Language Modeling as a Foundational Task

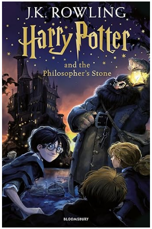
Compared to classification, which is somewhat a task-specific problem, language modeling is a more general task to be integrated into many NLP tasks.

Assigning probabilities to token sequences helps

- Machine translation
  $P(\textit{turn the camera off}) > P(\textit{put the camera out})$
- Speech recognition
  $P(\textit{a tomato garden}) > P(\textit{a tornado garden})$
- Grammatical error correction
  $P(\textit{about fifteen minutes}) > P(\textit{about fifteen minuets})$

Language Models: Data Matters

$$\Theta^* = \arg\max_{\Theta} \sum_{i=1}^{N} \log P_{\Theta}(\mathbf{x}_i)$$



WIKIPEDIA
The Free Encyclopedia

Language models highly depend on their training data that define the population distribution.

Language Modeling: From Word to Sequence Probability

Goal: compute the probability of a sequence of tokens.

$$P(\mathbf{x}_{1:T}) = P(x_1, x_2, \ldots, x_T)$$

Related task: compute the probability of the next word.

$$P(x_T | x_{1:T-1}) = P(x_T | x_1, x_2, \ldots, x_{T-1})$$

From a high-level modeling perspective,

- **Autoregressive language models**: compute $P(x_T | x_1, x_2, \ldots, x_{T-1})$.
- **Masked language models**: compute $P(x_k | x_1, \ldots, x_{k-1}, x_{k+1}, \ldots)$ with some tokens masked.

From a methodological perspective,

- **Count-based** language models: n-gram models.
- **Neural** language models: RNNs, LSTMs, Transformers.

Autoregressive Language Models

Recall the chain rule of probability:

$$P(A, B) = P(A)P(B \mid A)$$
$$P(A, B, C) = P(A)P(B \mid A)P(C \mid A, B)$$

Applying it to sequences of tokens:

$$\begin{aligned} &P(x_1, x_2, \ldots, x_T) \\ =&P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2) \ldots P(x_T \mid x_1, x_2, \ldots, x_{T-1}) \end{aligned}$$

An **autoregressive language model** computes the conditional probability $P(x_T \mid x_1, x_2, \ldots, x_{T-1})$.

Important detail: remember to model **sequence length** – a special token $\langle \texttt{EOS} \rangle$ is necessary in probabilistic terms!

Modeling Length: The End-of-Sequence Token

A language model assigns **probabilities** to token sequences $\mathbf{x}$ at a desired granularity (e.g., sentences, paragraphs, documents).

Given that granularity, $\mathbf{x}$ can be of any length.

To form a well-defined probability distribution, we need to have

$$\sum_{\mathbf{x}} P(\mathbf{x}) = 1.$$

Sequence length is modeled by including a special token $\langle \texttt{EOS} \rangle$.

$P(\langle \texttt{EOS} \rangle \mid \mathbf{x})$ denotes the **stop probability**.

Instead of calculating $P(x_1, x_2, \ldots, x_T)$, we calculate $P(x_1, x_2, \ldots, x_T, \langle \texttt{EOS} \rangle)$ as the sequence probability.

Modeling Length: The End-of-Sequence Token

What if we don't have the $\langle \texttt{EOS} \rangle$ token?

Recall our autoregressive language model calculates

$$P(x_1, x_2, \ldots, x_T) = P(x_1)P(x_2 \mid x_1) \ldots \underbrace{P(x_T \mid x_1, x_2, \ldots, x_{T-1})}_{\text{probability,} \in [0,1]}$$

If there is no $\langle \texttt{EOS} \rangle$ token

$$\begin{aligned} P(x_1, \ldots, x_T) &= P(x_1, \ldots x_{T-1})P(x_T \mid x_1, \ldots, x_{T-1}) \\ &\leq P(x_1, \ldots, x_{T-1}) \end{aligned}$$

$$P(\textit{The cat is cute.}) \leq P(\textit{The cat is})$$

Modeling Length: The End-of-Sequence Token

What if we don't have the ⟨EOS⟩ token?

Recall our autoregressive language model calculates

$$P(x_1, x_2, \ldots, x_T) = P(x_1)P(x_2 \mid x_1)\ldots \underbrace{P(x_T \mid x_1, x_2, \ldots, x_{T-1})}_{\text{probability}, \in [0,1]}$$

If there is no ⟨EOS⟩ token

$$(\text{length } T = 1) \quad \sum_{\mathbf{x}} P(\mathbf{x}_{1:T}) = \sum_{x_1 \in V} P(x_1) = 1$$

$$(\text{length } T = 2) \quad \sum_{\mathbf{x}} P(\mathbf{x}_{1:T}) = \sum_{x_1 \in V} \sum_{x_2 \in V} P(x_1)P(x_2 \mid x_1)$$

$$= \sum_{x_1 \in V} P(x_1) \left(\sum_{x_2 \in V} P(x_2 \mid x_1) \right) = 1$$

V: vocabulary.

Modeling Length: The End-of-Sequence Token

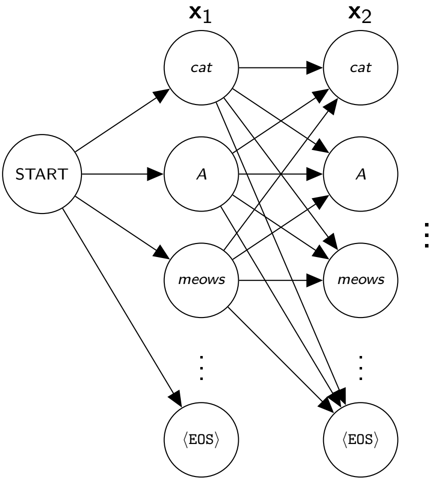
If we have the $\langle \texttt{EOS} \rangle$ token, the sum of sequence probability becomes

$$\sum_{\mathbf{x}} P(\mathbf{x}, \langle \texttt{EOS} \rangle) = 1$$

Idea for proof: once you reach the $\langle \texttt{EOS} \rangle$ token after sampling $\mathbf{x}_{1:T}$, certain probability mass is taken away—longer sequences that use $\mathbf{x}_{1:T}$ share the remaining probability mass.

Practice: complete the proof.

Modeling Length: The End-of-Sequence Token



$$P(\textit{meows}, \langle \texttt{EOS} \rangle) = P(\textit{meows}) P(\langle \texttt{EOS} \rangle \mid \textit{meows})$$

Each edge represents a (conditional) probability term after factorization.

N-Gram Language Models: The Markov Assumption

Q: Suppose we have a vocabulary size $|V| = 50\text{K}$, how many sequences of length T can we have?

A: $|V|^T$, which could be extremely large when $T \geq 3$.

Counting-based methods cannot efficiently model the conditional probability $P(x_T \mid x_1, x_2, \ldots, x_{T-1})$ when n goes large.

$$\begin{aligned} & P(\textit{The cat is cute.}) \\ = & P(\textit{The})P(\textit{cat} \mid \textit{The})P(\textit{is} \mid \textit{The cat}) \\ & P(\textit{cute} \mid \textit{The cat is})P(. \mid \textit{The cat is cute}) \end{aligned}$$

The **Markov assumption**: the probability of a token only depends on the previous $n-1$ tokens ($n << \text{sequence length } T$).

[Andrey Markov]

N-Gram Language Models: The Markov Assumption

In other words, the **Markov assumption** assumes independence of a token from distant history, conditioning on its close history.

$$P(x_i \mid x_1, x_2, \ldots, x_{i-1}) \approx P(x_i \mid \underbrace{x_{i-n+1}, x_{i-n+2}, \ldots, x_{i-1}}_{\text{always } n-1 \text{ entries}})$$

We can estimate the conditional probability $P(x_i \mid x_{i-n+1}, x_{i-n+2}, \ldots, x_{i-1})$ by counting the occurrences of n-grams:

$$P(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) = \frac{\text{count}(x_{i-n+1}, \ldots, x_{i-1}, x_i)}{\text{count}(x_{i-n+1}, \ldots, x_{i-1})}$$

Common N-Gram Language Models

- **Unigram language models** (n=1):

$$P(\mathbf{x}) = P(x_1)P(x_2)\dots P(x_T)$$
$$P(\textit{This is a cute cat}) = P(\textit{This})P(\textit{is})P(\textit{a})P(\textit{cute})P(\textit{cat})$$

  The `Sentencepiece` tokenizer (Kudo et al., 2018) uses this method to model text probability.
  Caveat: there is no way to have the $\langle \texttt{EOS} \rangle$ fix for unigram LMs.

- **Bigram language models** (n=2):

$$P(\mathbf{x}) = P(x_1) \sum_{i=2}^{T} P(x_i \mid x_{i-1})$$
$$P(\textit{This is a cute cat}) = P(\textit{This})P(\textit{is} \mid \textit{This})P(\textit{a} \mid \textit{is})P(\textit{cute} \mid \textit{a}) P(\textit{cat} \mid \textit{cute})P(\langle \texttt{EOS} \rangle \mid \textit{cat})$$

- **N-Gram language models** (n>2): similar to bigram models—should be paired with sparse techniques to store the probabilities.

Sample Sentences from Unigram and Bigram LMs

Both trained on financial news.

Model 1: Unigram LM

fifth an of futures the an incorporated a a the inflation most dollars quarter in is mass thrift did eighty said hard 'm july bullish that or limited the

Model 2: Bigram LM

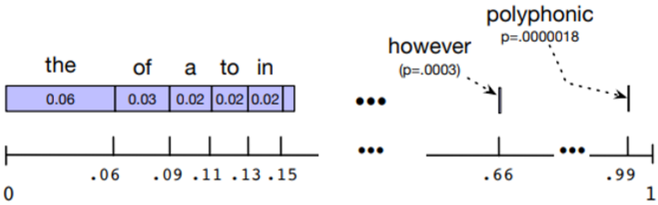
texaco rose one in this issue is pursuing growth in a boiler house said mr. gurria mexico 's motion control proposal without permission from five hundred fifty five yen outside new car parking lot of the agreement reached this would be a record november

Generating from a Language Model

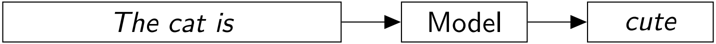
Taking bigram LMs as an example,

- Generate the first word $w_1 \sim P(w_1)$.
- Generate the second word $w_2 \sim P(w_2 \mid w_1)$.
- Generate the third word $w_3 \sim P(w_3 \mid w_1, w_2) = P(w_3 \mid w_2)$.
- ...
- Repeat until the $\langle \texttt{EOS} \rangle$ token is generated.

Recap: sampling from a distribution.



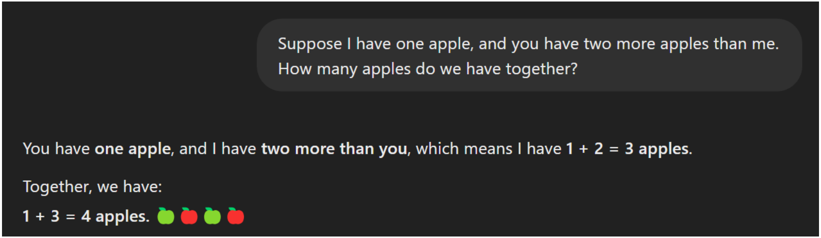
Neural Autoregressive LMs



This can be treated as $|V|$-way classification problems, with regular classification approaches.

Key idea: generate one token at a time.

Compared to n-gram LMs, Transformer-based LMs can handle much longer dependencies and generate coherent text.



Neural Autoregressive LMs: Training

Suppose training examples are drawn from an i.i.d. distribution.

Objective: maximize the (log) likelihood of the training data, which can be broken down into token-level probabilities.

$$\begin{aligned}\boldsymbol{\Theta}^* &= \arg\max_{\boldsymbol{\Theta}} \sum_{i=1}^{N} \log P_{\boldsymbol{\Theta}}(\mathbf{x}_i) \\ &= \arg\max_{\boldsymbol{\Theta}} \sum_{i=1}^{N} \sum_{t=1}^{T_i} \log P_{\boldsymbol{\Theta}}(x_{i,t} \mid x_{i,1}, \ldots, x_{i,t-1})\end{aligned}$$

Recap: Unified View of NLP

$$\arg\max_{y} score(s, y; \mathbf{\Theta})$$

s: input text, y: output, $\mathbf{\Theta}$: model parameters.

Past lectures: text classification, with y being a class label.

These two lectures: language models, with y being a word and s being the context.

- From the classification perspective, this is a natural extension of classification.
- From the word embeddings perspective, we are now allowed to use more complex models $score(s, y; \mathbf{\Theta})$.

Generating Text from Language Models

Given a well-trained language model $P_{\Theta}(x_t \mid x_1, \ldots, x_{t-1})$, how do we generate text?

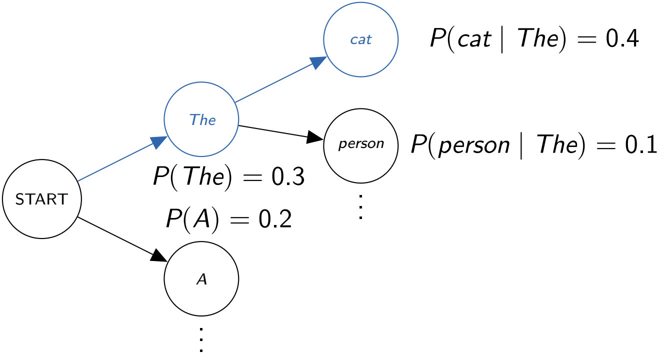
At each time step, we have several options:

- **Greedy decoding**: choose the token with the highest probability.
- **Beam search**: keep track of the top-k hypotheses.
- **Sampling**: sample from the distribution.
- **Top-k sampling**: sample from the top-k tokens with the highest probability.
- **Nucleus sampling (top-p) sampling**: sample from the smallest set of tokens whose cumulative probability exceeds a threshold p.

Greedy Decoding

At each time step, choose the token with the highest probability.

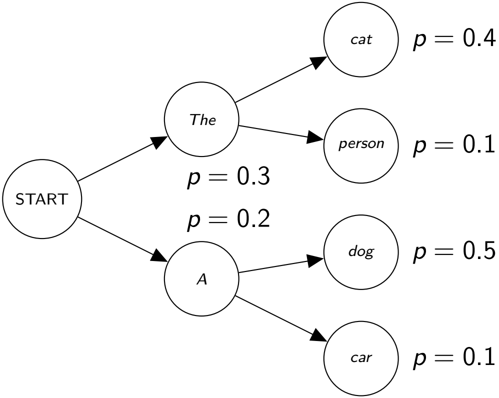
Repeat until the ⟨EOS⟩ token is generated, or it reaches a maximum length.



Beam Search

At each time step, keep track of the top-k hypotheses.

Repeat until the ⟨EOS⟩ token is generated, or it reaches a maximum length.



Example
(Beam size = 2)

Step 1:

- The (0.3)
- A (0.2)

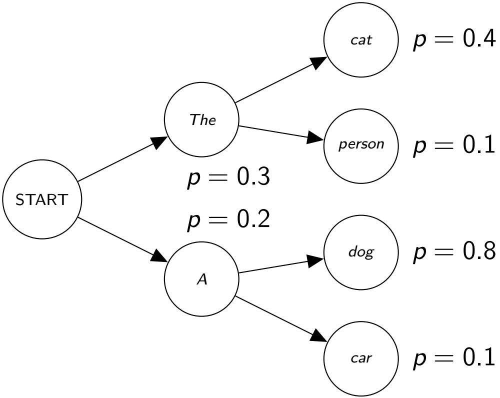
Step 2:

- The cat (0.12)
- A dog (0.1)

Greedy Decoding vs. Beam Search

Q: Which one gives a higher probability among all 2-token sequences, greedy decoding or beam search ($k = 2$)?

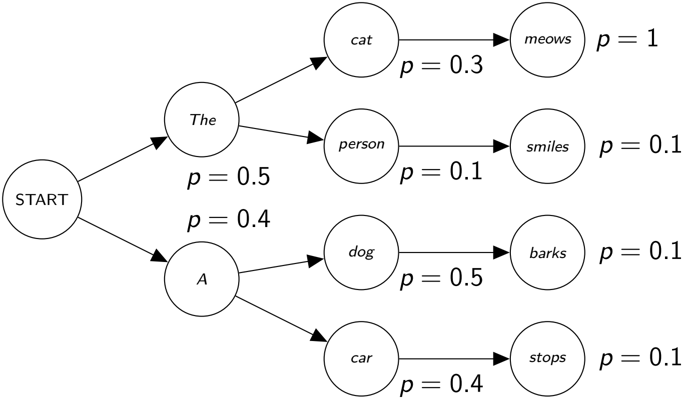
A: Beam search (*A dog*, $P = 0.16$).



Greedy Decoding vs. Beam Search

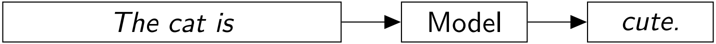
Q: Which one gives a sequence with higher probability among all 3-token sentences, greedy decoding or beam search ($k = 2$)?

A: Greedy decoding (*The cat meows*, $P = 0.15$).

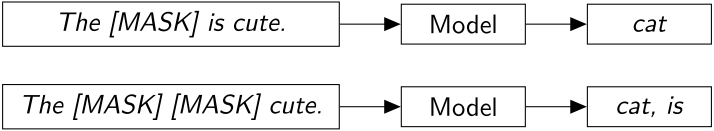


Language Modeling: Summary

Autoregressive language modeling (e.g., GPT, Radford et al., 2018):



Masked language modeling (BERT, Devlin et al., 2019):



Sampling

A language model defines a probability distribution over the vocabulary at each time step, which we can sample from.

In addition to direct sampling, there are several advanced strategies to sample from the distribution:

Top-k sampling

Avoid sampling from the tail of the distribution.

Top-p (Nucleus) sampling
(Holtzman et al., 2019)

Another way to define the tail of the distribution.

Evaluating Language Models

Extrinsic (task-based) evaluation: use the language model as a component in a downstream task, and see if the performance improves.

Downsides:

- Can be time-consuming.
- The performance may be affected by how LMs are used.

Intrinsic evaluation: compute and compare the probability on held-out data, where **perplexity** is the standard metric.

Downsides:

- May not correlate well with downstream task performance.

Perplexity of Held-Out Data

Log-probability of held-out data $\mathcal{X}$ with model $P_{\mathbf{\Theta}}$:

$$\log P_{\mathbf{\Theta}}(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} \log_2 P_{\mathbf{\Theta}}(\mathbf{x})$$

Divide by the number of tokens (including the $\langle \texttt{EOS} \rangle$ token) to get the average log-probability per token:

$$\ell = \text{Avg } \log P_{\mathbf{\Theta}}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \log_2 P_{\mathbf{\Theta}}(\mathcal{X})$$

$$\mathit{Perplexity}(\mathcal{X}; \mathbf{\Theta}) = 2^{-\ell} = 2^{-\frac{1}{|\mathcal{X}|} \log_2 P_{\mathbf{\Theta}}(\mathcal{X})} = P_{\mathbf{\Theta}}(\mathcal{X})^{-\frac{1}{|\mathcal{X}|}}$$

ℓ: token-level cross-entropy loss.

Higher the probability of the held-out data means... it's less perplexing to the model.

Next

Masked Language Models, Sequence Labeling