DSC250: Advanced Data Mining

Language Models

Zhiting Hu

Lecture 8, October 24, 2023

UC San Diego

HALICIOĞLU DATA SCIENCE INSTITUTE

Outline

- Variational Inference (quick overview)
- Language Models

Recap: EM Algorithm

- Observed variables $\boldsymbol{x}$, latent variables $\boldsymbol{z}$
- To learn a model $p(\boldsymbol{x}, \boldsymbol{z}|\theta)$, we want to maximize the marginal log-likelihood

$$\ell(\theta; \boldsymbol{x}) = \log p(\boldsymbol{x}|\theta) = \log \sum_{z} p(\boldsymbol{x}, \boldsymbol{z}|\theta)$$

  - But it's too difficult
- EM algorithm:
  - maximize a lower bound of $\ell(\theta; \boldsymbol{x})$
  - Or equivalently, minimize an upper bound of $-\ell(\theta; \boldsymbol{x})$
- Key equation:

Evidence Lower Bound (ELBO)

$$\ell(\theta; \boldsymbol{x}) = \boxed{\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}|\theta)}{q(\boldsymbol{z}|\boldsymbol{x})}\right]} + \mathrm{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}|\boldsymbol{x}, \theta)\big)$$

$$= -\boxed{F(q, \theta)} + \mathrm{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}|\boldsymbol{x}, \theta)\big)$$

Variational free energy

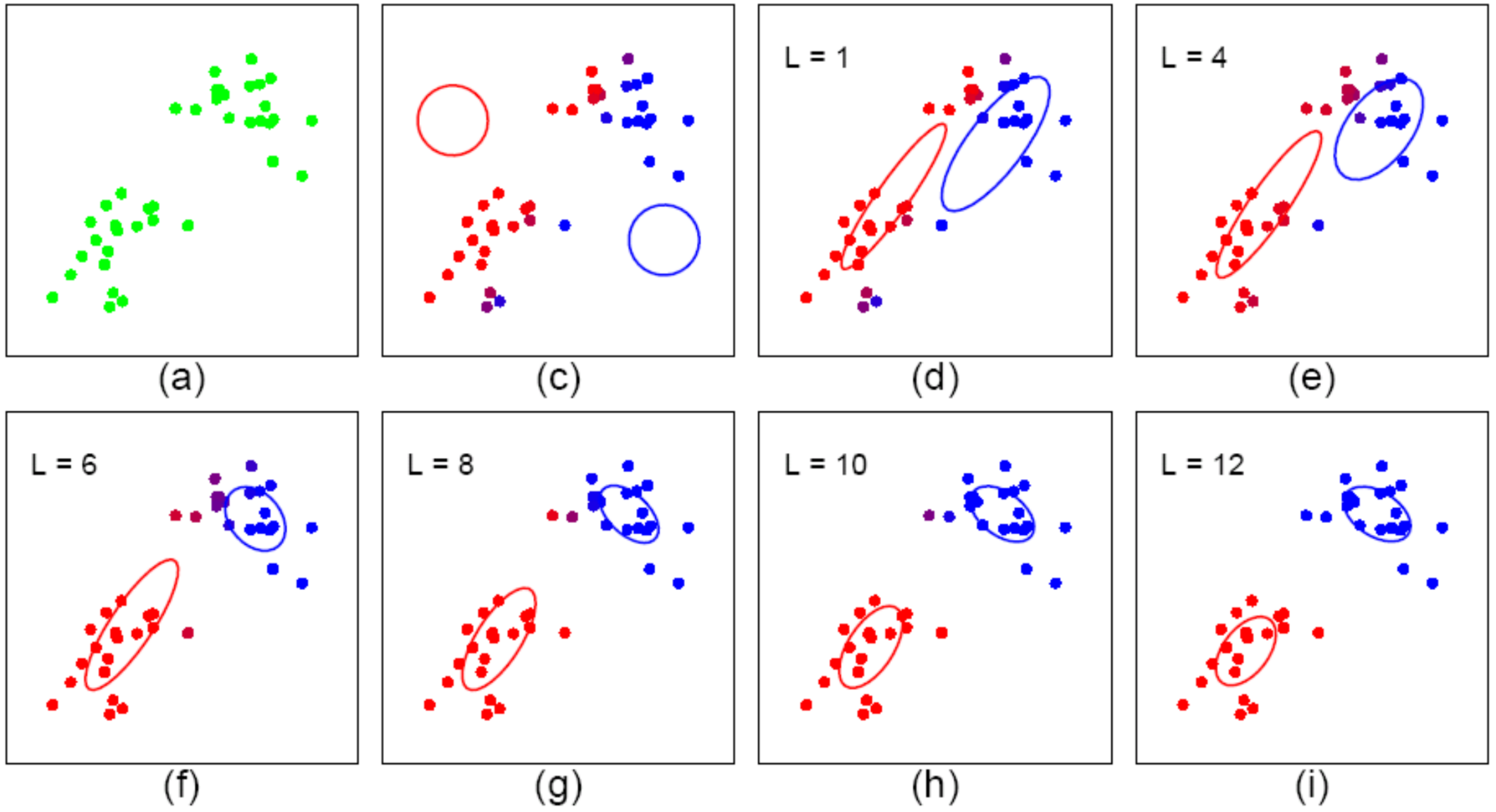
Recap: EM Algorithm

- The EM algorithm is coordinate-decent on $F(q, \theta)$
  - E-step: $q^{t+1} = \arg\min_q F\left(q, \theta^t\right) = p(\boldsymbol{z}|\boldsymbol{x}, \theta^t)$
    - the posterior distribution over the latent variables given the data and the current parameters
  - M-step: $\theta^{t+1} = \arg\min_\theta F\left(q^{t+1}, \theta^t\right) = \text{argmax}_\theta \sum_z q^{t+1}(\boldsymbol{z}|\boldsymbol{x}) \log p(\boldsymbol{x}, \boldsymbol{z}|\theta)$

$$\boxed{\begin{aligned} \ell(\theta; \boldsymbol{x}) &= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}|\theta)}{q(\boldsymbol{z}|\boldsymbol{x})}\right] + \text{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}|\boldsymbol{x}, \theta)\big) \\ &= -F(q, \theta) + \text{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}|\boldsymbol{x}, \theta)\big) \end{aligned}}$$

Example: Gaussian Mixture Models (GMMs)

- Start: "guess" the centroid μ_k and covariance Σ_k of each of the K clusters
- Loop:



Variational Inference (quick overview)

Content adapted from CMU 10-708 Spring 2017

Inference

- Given a model, the goals of inference can include:
  - Computing the likelihood of observed data $p(\boldsymbol{x}^*)$
  - Computing the marginal distribution over a given subset of variables in the model $p(\boldsymbol{x}_A)$
  - Computing the conditional distribution over a subsets of nodes given a disjoint subset of nodes $p(\boldsymbol{x}_A|\boldsymbol{x}_B)$
  - Computing a mode of the density (for the above distributions) $\operatorname{argmax}_{\boldsymbol{x}} p(\boldsymbol{x})$
  - ….

Variational Inference

- Observed variables $\boldsymbol{x}$, latent variables $\boldsymbol{z}$
- Variational (Bayesian) inference, a.k.a. **variational Bayes**, is most often used to approximately infer the conditional distribution over the latent variables given the observations (and parameters)
    - i.e., the **posterior distribution** over the latent variables

$$p(\boldsymbol{z}|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{z}, \boldsymbol{x}|\theta)}{\sum_z \; p(\boldsymbol{z}, \boldsymbol{x}|\theta)}$$

Motivating Example

- Why do we often need to use an approximate inference methods (such as variational Bayes) to compute the posterior distribution?

- It’s because we cannot directly compute the posterior distribution for many interesting models
  - I.e. the posterior density is in an intractable form (often involving integrals) which cannot be easily analytically solved.

Variational Inference

The main idea behind variational inference:

- Choose a family of distributions over the latent variables $z_{1:m}$ with its own set of variational parameters ν, i.e.

$$q(z_{1:m}|\nu)$$

- Then, we find the setting of the parameters that makes our approximation q closest to the posterior distribution.
    - This is where optimization algorithms come in.

- Then we can use q with the fitted parameters in place of the posterior.
    - E.g. to form predictions about future data, or to investigate the posterior distribution over the hidden variables, find modes, etc.

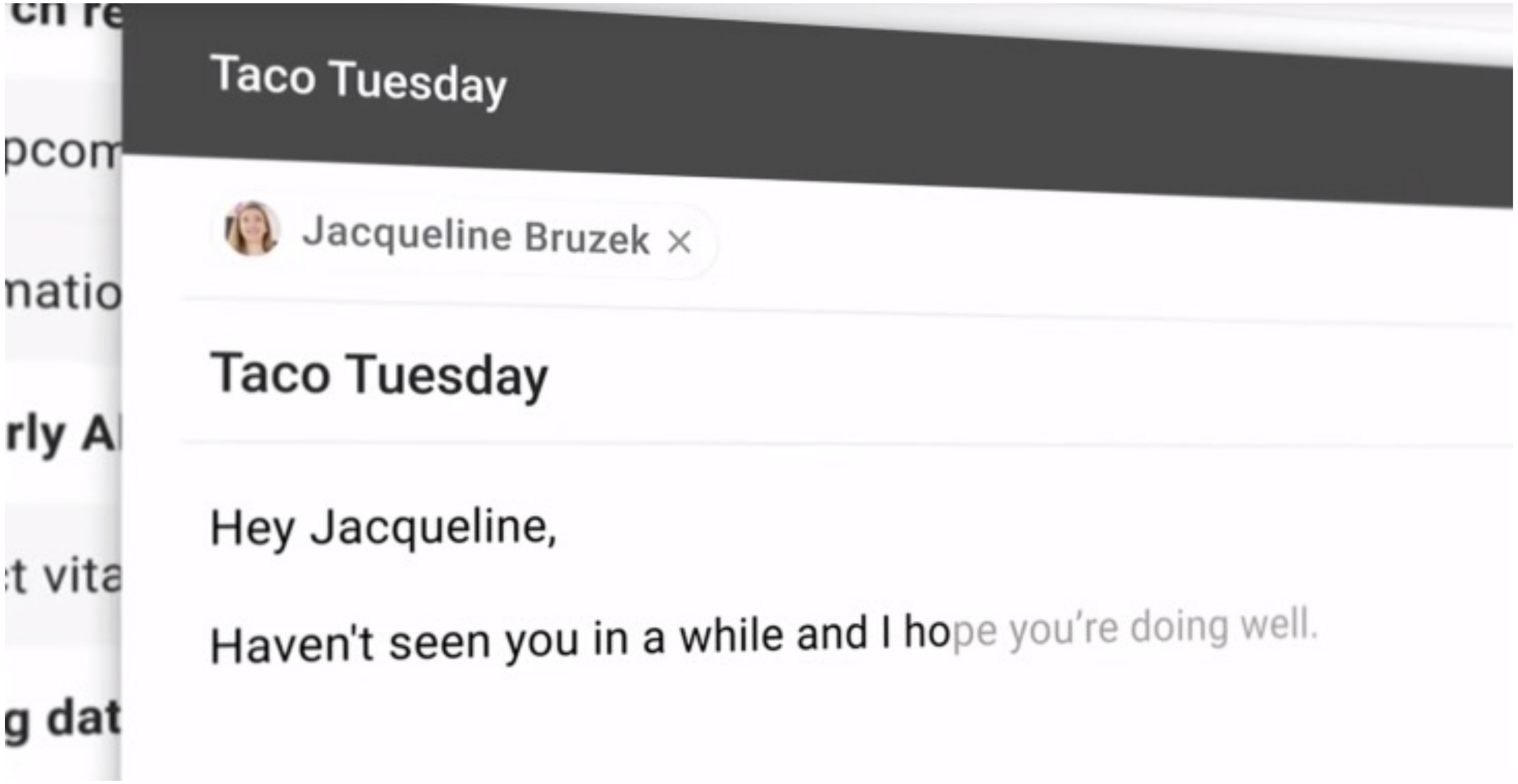
Language Models

Outline

- N-gram language models
- Neural language models
- Neural architectures (in general)

Motivations of Language Models

- Generation



Taco Tuesday

Jacqueline Bruzek ×

Taco Tuesday

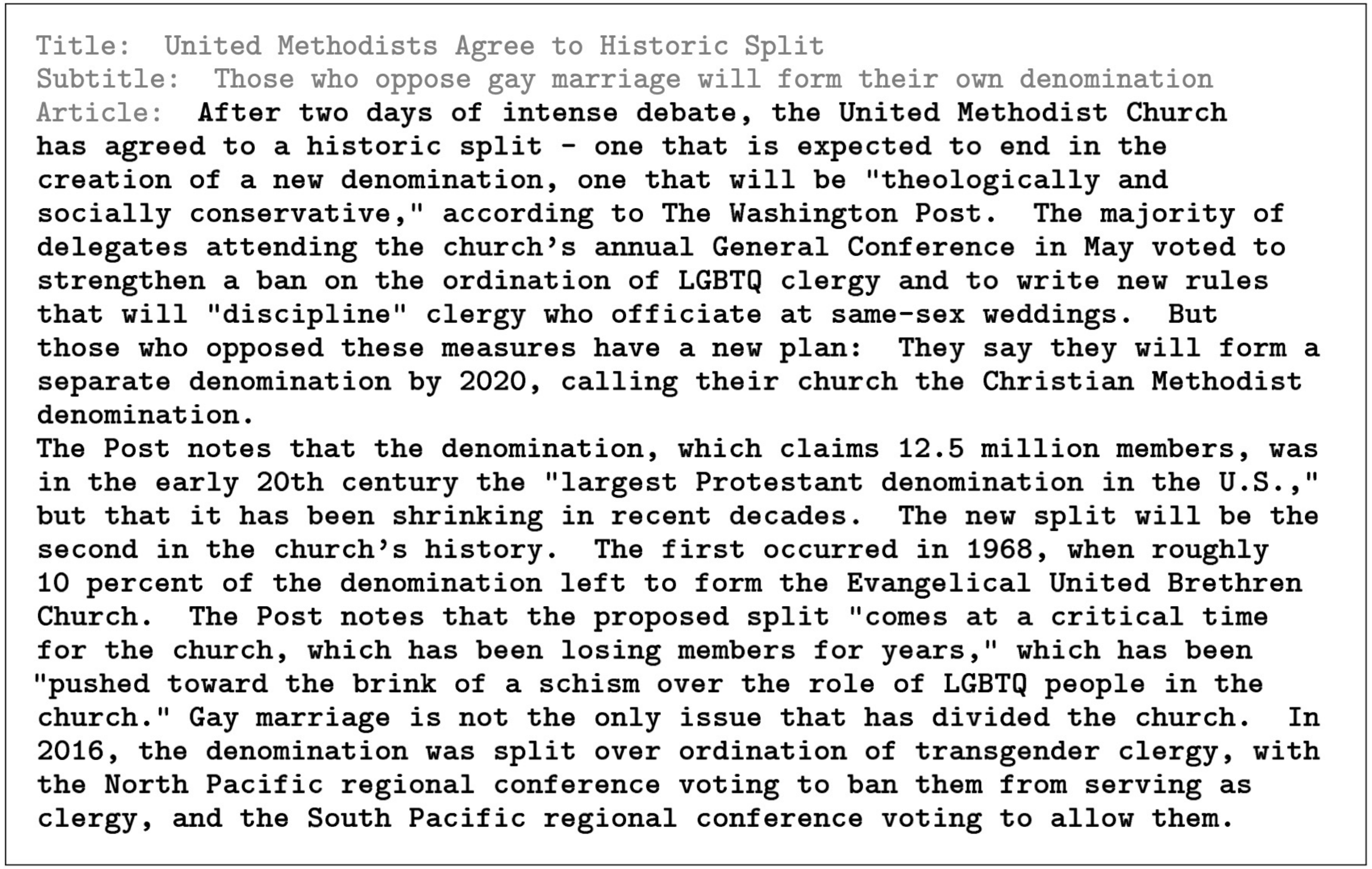
Hey Jacqueline,

Haven't seen you in a while and I hope you're doing well.

Email auto-completion

Motivations of Language Models

- Generation

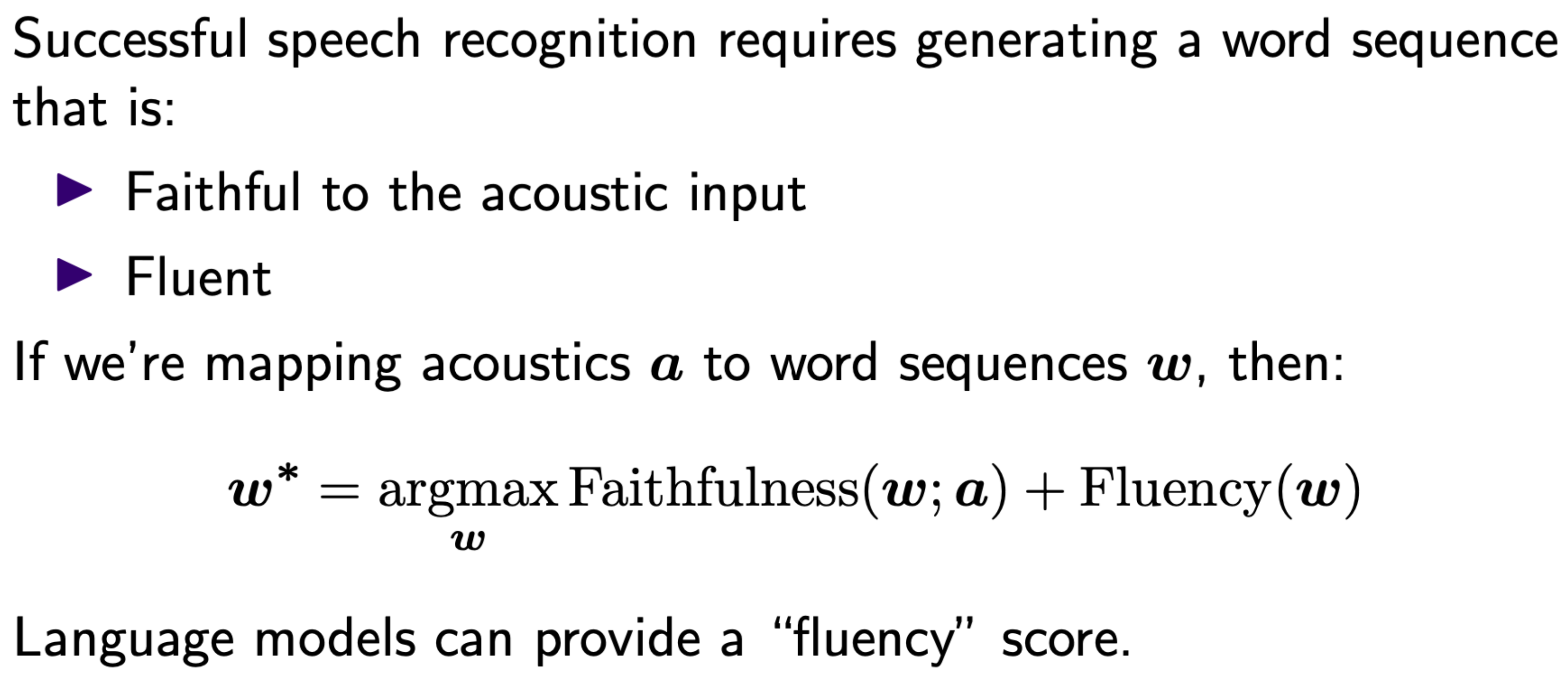


```
Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: After two days of intense debate, the United Methodist Church
has agreed to a historic split - one that is expected to end in the
creation of a new denomination, one 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.
The Post notes that the denomination, which claims 12.5 million members, was
in the early 20th century the "largest Protestant denomination in the U.S.,"
but that it has been shrinking in recent decades. The new split will be the
second in the church's history. The first occurred in 1968, when roughly
10 percent of the denomination left to form the Evangelical United Brethren
Church. The Post notes that the proposed split "comes at a critical time
for the church, which has been losing members for years," which has been
"pushed toward the brink of a schism over the role of LGBTQ people in the
church." Gay marriage is not the only issue that has divided the church. In
2016, the denomination was split over ordination of transgender clergy, with
the North Pacific regional conference voting to ban them from serving as
clergy, and the South Pacific regional conference voting to allow them.
```

Figure 3.14: The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%).

Motivations of Language Models

- Evaluation of language fluency
  - Ex:



Successful speech recognition requires generating a word sequence that is:

- Faithful to the acoustic input
- Fluent

If we're mapping acoustics $\boldsymbol{a}$ to word sequences $\boldsymbol{w}$, then:

$$\boldsymbol{w}^* = \operatorname*{argmax}_{\boldsymbol{w}} \text{Faithfulness}(\boldsymbol{w}; \boldsymbol{a}) + \text{Fluency}(\boldsymbol{w})$$

Language models can provide a "fluency" score.

[Courtesy: UW CSE 447 by Noah Smith]

Motivations of Language Models

- Evaluation of language fluency
  - Ex: Other tasks that have text (or speech) as output:
    - ▶ translation from one language to another
    - ▶ conversational systems
    - ▶ document summarization
    - ▶ image captioning
    - ▶ optical character recognition
    - ▶ spelling and grammar correction

    If we're mapping inputs $\boldsymbol{i}$ to word sequences $\boldsymbol{w}$, then:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}} \operatorname{Faithfulness}(\boldsymbol{w}; \boldsymbol{i}) + \operatorname{Fluency}(\boldsymbol{w})$$

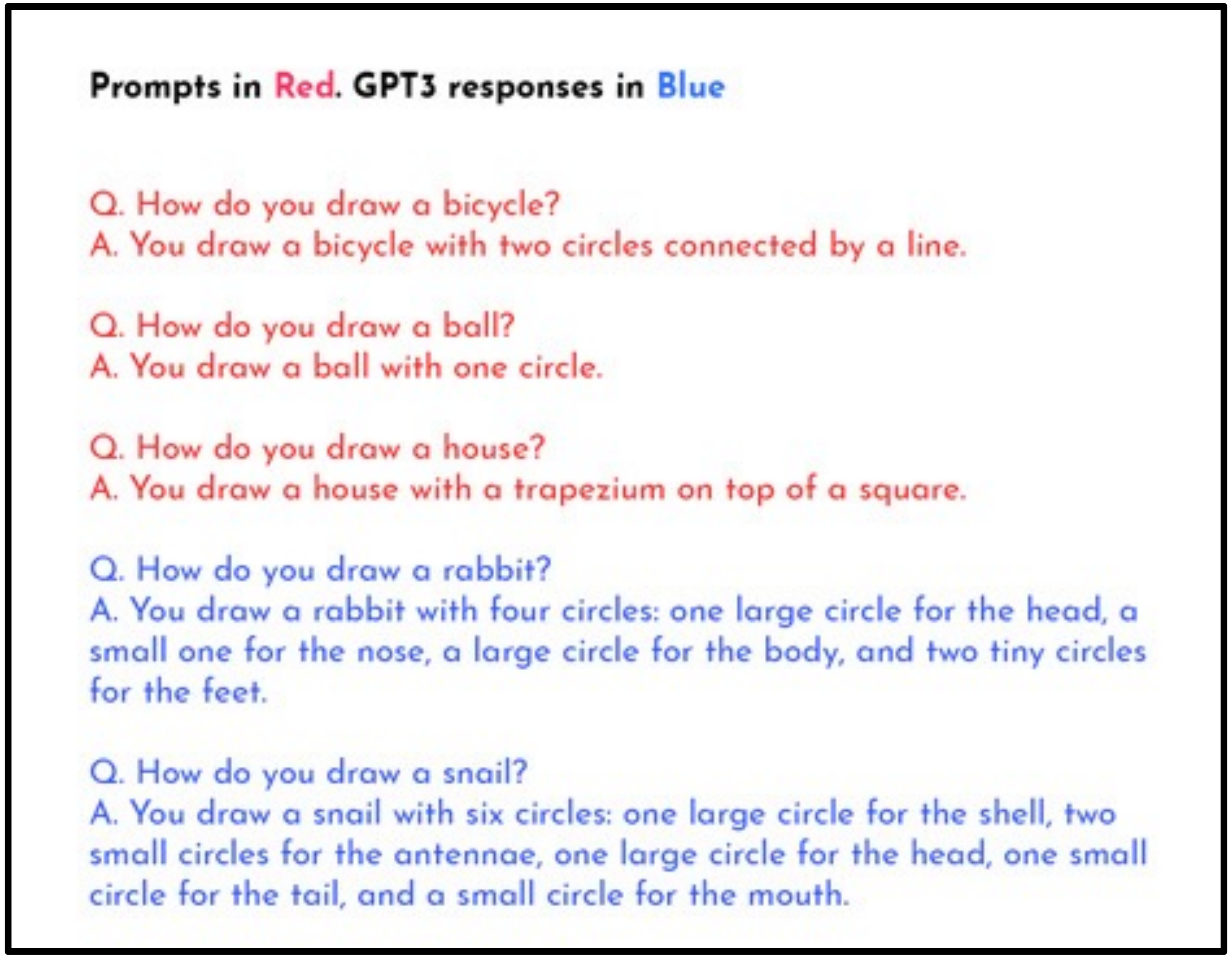
    Language models can provide a "fluency" score.

[Courtesy: UW CSE 447 by Noah Smith]

Brown et al., 2020 "Language Models Are Few-Shot Learners”

Motivations of Language Models

- Few-shot prediction (e.g., GPT3)



Prompts in Red. GPT3 responses in Blue

Q. How do you draw a bicycle?
A. You draw a bicycle with two circles connected by a line.

Q. How do you draw a ball?
A. You draw a ball with one circle.

Q. How do you draw a house?
A. You draw a house with a trapezium on top of a square.

Q. How do you draw a rabbit?
A. You draw a rabbit with four circles: one large circle for the head, a small one for the nose, a large circle for the body, and two tiny circles for the feet.

Q. How do you draw a snail?
A. You draw a snail with six circles: one large circle for the shell, two small circles for the antennae, one large circle for the head, one small circle for the tail, and a small circle for the mouth.

[Nurecas.com]

Notations

- $\mathcal{V}$ is a finite set of (discrete) symbols (words or characters); $V = |\mathcal{V}|$
- $\mathcal{V}^*$ is the (infinite) set of sequences of symbols from $\mathcal{V}$

[Courtesy: UW CSE 447 by Noah Smith]

Notations

- $\mathcal{V}$ is a finite set of (discrete) symbols (words or characters); $V = |\mathcal{V}|$
- $\mathcal{V}^*$ is the (infinite) set of sequences of symbols from $\mathcal{V}$
- In language modeling, we imagine a sequence of random variables $X_1, X_2, \ldots$ that continues until some X_n takes the value "⯃" (a special end-of-sequence symbol).
- $\mathcal{V}^\dagger$ is the (infinite) set of sequences of $\mathcal{V}$ symbols, with a single ⯃, which is at the end.

[Courtesy: UW CSE 447 by Noah Smith]

The Language Modeling Problem

- Input: training data $\boldsymbol{x} = (x_1, x_2, \dots, x_N)$ in $\mathcal{V}^\dagger$
  - (assuming one instance $\boldsymbol{x}$ for simplicity of notations)
- Output: p: $\mathcal{V}^\dagger \rightarrow \mathbb{R}$
- Think of p as a measure of plausibility

[Courtesy: UW CSE 447 by Noah Smith]

Probabilistic Language Model

- We let p be a probability distribution, which means that

$$\forall \boldsymbol{x} \in \mathcal{V}^\dagger, p(\boldsymbol{x}) \geq 0$$

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

- Advantages:
    - Interpretability
    - We can apply the maximum likelihood principle to build a language model from data

Decomposing using the Chain Rule

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

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

Example:

Predict each word based on the "history"

$\boldsymbol{x}$ = *(I, like, this, movie, …)*

$$p(\boldsymbol{x}) = \cdots\, p_\theta(like \mid I)\; p_\theta(this \mid I, like)\, \cdots$$

Unigram Model: Empty History

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

$$\overset{\text{assumption}}{=} \prod_{i=1}^{N} p(X_i = x_i; \boldsymbol{\theta}) = \prod_{i=1}^{N} \theta_{x_i}$$

Multinomial distribution

[Courtesy: UW CSE 447 by Noah Smith]

Unigram Model: Empty History

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

Multinomial distribution

$$\overset{\text{assumption}}{=} \prod_{i=1}^{N} p(X_i = x_i; \boldsymbol{\theta}) = \prod_{i=1}^{N} \theta_{x_i}$$

Maximum likelihood estimate: for every $v \in \mathcal{V}$,

$$\theta_v^* = \frac{\sum_{i=1}^{N} \mathbf{1}\{x_i = v\}}{N}$$

$$= \frac{\text{count}_{\boldsymbol{x}}(v)}{N}$$

Example

The probability of

Presidents tell lies .

is:

$$p(X_1 = \text{Presidents}) \cdot p(X_2 = \text{tell}) \cdot p(X_3 = \text{lies}) \cdot p(X_4 = .) \cdot p(X_5 = \text{🛑})$$

In unigram model notation:

$$\theta_{\text{Presidents}} \cdot \theta_{\text{tell}} \cdot \theta_{\text{lies}} \cdot \theta_{.} \cdot \theta_{\text{🛑}}$$

Using the maximum likelihood estimate for $\boldsymbol{\theta}$, we could calculate:

$$\frac{\mathrm{count}_{\boldsymbol{x}}(\text{Presidents})}{N} \cdot \frac{\mathrm{count}_{\boldsymbol{x}}(\text{tell})}{N} \cdots \frac{\mathrm{count}_{\boldsymbol{x}}(\text{🛑})}{N}$$

Unigram Models: Assessment

Pros:

- Easy to understand
- Cheap
- Good enough for information retrieval (maybe)

Cons:

- Fixed, known vocabulary assumption
- "Bag of words" assumption is linguistically inaccurate
  - $p(\text{the the the the}) \gg p(\text{I want ice cream})$

n-gram Models

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

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

$$= \prod_{i=1}^{N} \theta_{x_i \mid \boldsymbol{x}_{i-\mathsf{n}+1:i-1}}$$

[Courtesy: UW CSE 447 by Noah Smith]

n-gram Models

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

$(\mathsf{n} - 1)$th-order Markov assumption $\equiv$ n-gram model

- Unigram model is the $\mathsf{n} = 1$ case
- For a long time, trigram models $(\mathsf{n} = 3)$ were widely used
- 5-gram models $(\mathsf{n} = 5)$ were common in MT for a time

[Courtesy: UW CSE 447 by Noah Smith]

n-gram Models

- Maximum likelihood estimate for the n-gram model's probability of v given a $(n - 1)$-length history $\boldsymbol{h}$

$$\begin{aligned}
\theta_{v|\boldsymbol{h}} &= p(X_i = v \mid \boldsymbol{X}_{i-\mathsf{n}+1:i-1} = \boldsymbol{h}) \\
&= \frac{p(X_i = v, \boldsymbol{X}_{i-\mathsf{n}+1:i-1} = \boldsymbol{h})}{p(\boldsymbol{X}_{i-\mathsf{n}+1:i-1} = \boldsymbol{h})} \\
&= \frac{\text{count}_{\boldsymbol{x}}(\boldsymbol{h}v)}{N} \Bigg/ \frac{\text{count}_{\boldsymbol{x}}(\boldsymbol{h})}{N} \\
&= \frac{\text{count}_{\boldsymbol{x}}(\boldsymbol{h}v)}{\text{count}_{\boldsymbol{x}}(\boldsymbol{h})}
\end{aligned}$$

[Courtesy: UW CSE 447 by Noah Smith]

Choosing n is a Balancing Act

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

As n gets larger:

- The number of parameters grows with $O(V^{\mathsf{n}})$.
- 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.

This is a beautiful illustration of the bias-variance tradeoff.

Other "tricks"

- Smoothing

  The game: prevent $\theta_{v|\boldsymbol{h}} = 0$ for any v and $\boldsymbol{h}$, while keeping $\sum_{\boldsymbol{x}} p(\boldsymbol{x}) = 1$ so that perplexity stays meaningful.

  - ▶ Simple method: add $\lambda > 0$ to every count (including counts of zero) before normalizing (the textbook calls this "Lidstone" smoothing)

- Dealing with Out-of-Vocabulary Terms
  - Define a special OOV or "unknown" symbol **unk**. Transform some (or all) rare words in the training data to **unk**.
  - Build a language model at the character level.
  - Some new methods use data-driven, deterministic tokenization schemes that segment some words into smaller parts to reduce the effective vocabulary size (Sennrich et al., 2016; Wu et al., 2016).

n-gram Models: Assessment

Pros:

- Easy to understand
- Cheap (with modern hardware; Lin and Dyer, 2010)
- Fine in some applications and when training data is scarce

Cons:

- Fixed, known vocabulary assumption
- Markov assumption is linguistically inaccurate
    - (But not as bad as unigram models!)
- Data sparseness problem

[Courtesy: UW CSE 447 by Noah Smith]

Neural Language Models

Neural Language Models

Instead of a lookup for a word and fixed-length history ($\theta_{v|\boldsymbol{h}}$), define a vector function:

$$p(X_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1}) = \mathbf{NN}(\mathbf{enc}(\boldsymbol{x}_{1:i-1}); \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ do the work of *encoding* the history and *transforming* it into a distribution over the next word.

Neural Language Models

Instead of a lookup for a word and fixed-length history ($\theta_{v|\boldsymbol{h}}$), define a vector function:

$$p(X_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1}) = \mathbf{NN}(\mathbf{enc}(\boldsymbol{x}_{1:i-1}); \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ do the work of *encoding* the history and *transforming* it into a distribution over the next word.
The transformation is described as a composed series of simple transformations or "layers."

Neural Network

Formally, it's a function $\mathbf{NN}$ from $\boldsymbol{\theta}$ (learned parameters) and inputs to outputs, all of which are real-valued vectors (or matrices, or tensors, or collections of them).

Almost always, $\mathbf{NN}$ is differentiable with respect to $\boldsymbol{\theta}$ and nonlinear with respect to the data input.

- "Nonlinear" means there does **not** exist a matrix $\mathbf{A}$ such that $\mathbf{NN}(\mathbf{v}; \boldsymbol{\theta}) = \mathbf{A}\mathbf{v}$, for all $\mathbf{v}$.

[Courtesy: UW CSE 447 by Noah Smith]

Neural Language Models

Instead of a lookup for a word and fixed-length history ($\theta_{v|\boldsymbol{h}}$), define a vector function:

$$p(X_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1}) = \mathbf{NN}(\mathbf{enc}(\boldsymbol{x}_{1:i-1}); \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ do the work of *encoding* the history and *transforming* it into a distribution over the next word.
The transformation is described as a composed series of simple transformations or "layers."

- We first map word histories $\boldsymbol{h}$ to vectors/matrices
- We interpret the output as $p(X_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{h})$

[Courtesy: UW CSE 447 by Noah Smith]

Two Key Components

- “Embedding” words as vectors
- Layering to increase capacity (i.e., the set of distributions that can be represented).

“One Hot” Vectors

Let $\mathbf{e}_i \in \mathbb{R}^V$ be the ith column of the identity matrix $\mathbf{I}$.

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}; \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}; \quad \ldots; \quad \mathbf{e}_V = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$\mathbf{e}_i$ is the “one hot” vector for the ith word in $\mathcal{V}$.

A neural language model starts by “looking up” each word by multiplying its one hot vector by a matrix $\underset{V \times d}{\mathbf{M}}$; $\mathbf{e}_v^\top \mathbf{M} = \mathbf{m}_v$, the “embedding” of v.

$\mathbf{M}$ becomes part of the parameters ($\boldsymbol{\theta}$).

Sequences of Word Vectors

Given a word sequence $\langle v_1, v_2, \ldots, v_k \rangle$, we transform it into a sequence of word vectors,

$$\mathbf{m}_{v_1}, \mathbf{m}_{v_2}, \ldots, \mathbf{m}_{v_k}$$

Adding Layers

- Neural networks are built by composing functions, a mix of
  - Affine, $\boldsymbol{v}' = \boldsymbol{W}\boldsymbol{v} + \boldsymbol{b}$ (note that the dimensionality of $\boldsymbol{v}$ and $\boldsymbol{v}'$ might be different)
  - Nonlinearity, e.g.,
    - rectified linear ("relu") units $v_i' = \max(0, v_i)$
    - elementwise hyperbolic tangent $v_i' = \tanh(v_i) = \dfrac{e^{v_i} - e^{-v_i}}{e^{v_i} + e^{-v_i}}$
    - softmax $v_i' = \exp\{ v_i \} / \sum_j \exp\{ v_j \}$
  - More complex components (composed of the above operations):
    - Convolutional layers
    - Recurrent NNs
    - Attention

Summary so far

- language models utilities
  - Generation, evaluation of fluency, few-shot prediction (GPT3), …
- N-gram language models
  - Unigram LM
  - N-gram LM
- Neural language models:
  - Embedding: one-hot vectors -> embedding vectors
  - Neural networks

Questions?