CS 533: Natural Language Processing

Language Modeling

Karl Stratos

Rutgers University

Motivation

How likely are the following sentences?

- `the dog barked`
- `the cat barked`
- `dog the barked`
- `oqc shgwqw#w 1g0`

Motivation

How likely are the following sentences?

- `the dog barked`

  "probability 0.1"

- `the cat barked`

  "probability 0.03"

- `dog the barked`

  "probability 0.00005"

- `oqc shgwqw#w 1g0`

  "probability 10^{-13}"

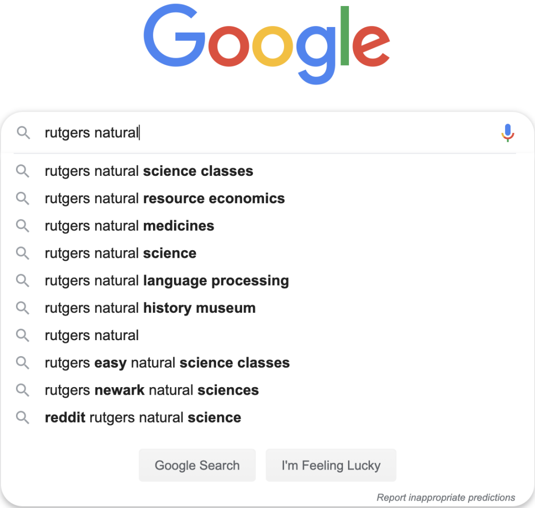
Language Model: Definition

A **language model** is a function that defines a probability distribution $p(x_1 \dots x_m)$ over all sentences $x_1 \dots x_m$.

Goal: Design a *good* language model, in particular

$$\begin{aligned} p(\texttt{the dog barked}) &> p(\texttt{the cat barked}) \\ &> p(\texttt{dog the barked}) \\ &> p(\texttt{oqc shgwqw\#w 1g0}) \end{aligned}$$

Language Models Are Everywhere



Text Generation with Modern Language Models

Custom prompt

Rutgers University has some of the best students in the country. They are particularly eager to learn about NLP.

GENERATE ANOTHER

Completion

Rutgers University has some of the best students in the country. They are particularly eager to learn about NLP. Their students seem to bring a much-needed diversity and inclusiveness to all disciplines. In fact, so pervasive is the research conducted here that only the Middle East comes close in attracting data.

So my question to you is, can you find a school that will create that same excitement and expectation? Can you inspire the spirit of discovery that you bring to all your classes? I think it will be very rewarding to learn from you, and I will be very happy if you become an alum! Thank you for reading

Try it yourself: https://talktotransformer.com/

Overview

Probability of a Sentence

n-Gram Language Models

- Unigram, Bigram, Trigram Models
- Estimation from Data
- Evaluation
- Smoothing

Log-Linear Language Models

Problem Statement

- We'll assume a finite **vocabulary** V (i.e., the set of all possible word types).
- Sample space: $\Omega = \{x_1 \ldots x_m \in V^m : \; m \geq 1\}$
- Task: Design a function p over Ω such that

$$\begin{aligned} p(x_1 \ldots x_m) &\geq 0 \qquad\qquad \forall x_1 \ldots x_m \in \Omega \\ \sum_{x_1 \ldots x_m \in \Omega} p(x_1 \ldots x_m) &= 1 \end{aligned}$$

- What are some challenges?

Challenge 1: Infinitely Many Sentences

- Can we "break up" the probability of a sentence into probabilities of individual words?

- **Yes**: Assume a *generative process*.

- We may assume that each sentence $x_1 \dots x_m$ is generated as

  (1) x_1 is drawn from $p(\cdot)$,
  (2) x_2 is drawn from $p(\cdot|x_1)$,
  (3) x_3 is drawn from $p(\cdot|x_1, x_2)$,
  $\dots$
  (m) x_m is drawn from $p(\cdot|x_1, \dots, x_{m-1})$,
  $(m+1)$ x_{m+1} is drawn from $p(\cdot|x_1, \dots, x_m)$.

  where $x_{m+1} = \texttt{STOP}$ is a special token at the end of every sentence.

Justification of the Generative Assumption

By the **chain rule**,

$$p(x_1 \dots x_m \, \texttt{STOP}) = p(x_1) \times p(x_2|x_1) \times p(x_3|x_1, x_2) \times \cdots$$
$$\cdots \times p(x_m|x_1, \dots, x_{m-1}) \times p(\texttt{STOP}|x_1, \dots, x_m)$$

Thus we have solved the first challenge.

- Sample space = *finite* V
- The model still defines a proper distribution over all sentences.

(Does the generative process need to be left-to-right?)

STOP Symbol

Ensures that there is probabilty mass left for longer sentences

Probabilty mass of sentences with length ≥ 1

$$1 - \underbrace{\sum_{x \in V} p(\texttt{STOP})}_{P(X_1 = \texttt{STOP}) = 0} = 1$$

Probabilty mass of sentences with length ≥ 2

$$1 - \underbrace{\sum_{x \in V} p(x\ \texttt{STOP})}_{P(X_2 = \texttt{STOP})} > 0$$

Probabilty mass of sentences with length ≥ 3

$$1 - \underbrace{\sum_{x \in V} p(x\ \texttt{STOP})}_{P(X_2 = \texttt{STOP})} - \underbrace{\sum_{x, x' \in V} p(x\ x'\ \texttt{STOP})}_{P(X_3 = \texttt{STOP})} > 0$$

Challenge 2: Infinitely Many Distributions

Under the generative process, we need infinitely many conditional word distributions:

$$
\begin{aligned}
p(x_1) \qquad & \qquad \forall x_1 \in V \\
p(x_2|x_1) \qquad & \qquad \forall x_1, x_2 \in V \\
p(x_3|x_1, x_2) \qquad & \qquad \forall x_1, x_2, x_3 \in V \\
p(x_4|x_1, x_2, x_3) \qquad & \qquad \forall x_1, x_2, x_3, x_4 \in V \\
\vdots \qquad & \qquad \vdots
\end{aligned}
$$

Now our goal is to redesign the model to have only a **finite, compact** set of associated values.

Overview

Probability of a Sentence

n-Gram Language Models

- Unigram, Bigram, Trigram Models
- Estimation from Data
- Evaluation
- Smoothing

Log-Linear Language Models

Independence Assumptions

X is **independent of** Y if

$$P(X = x|Y = y) = P(X = x)$$

X is **conditionally independent of** Y **given** Z if

$$P(X = x|Y = y, Z = z) = P(X = x|Z = z)$$

Can you think of such X, Y, Z?

Unigram Language Model

Assumption. A word is independent of all previous words:

$$p(x_i | x_1 \dots x_{i-1}) = p(x_i)$$

That is,

$$p(x_1 \dots x_m) = \prod_{i=1}^{m} p(x_i)$$

Number of parameters: $O(|V|)$

Not a very good language model:

$$p(\texttt{the dog barked}) = p(\texttt{dog the barked})$$

Bigram Language Model

Assumption. A word is independent of all previous words conditioning on the preceding word:

$$p(x_i|x_1 \ldots x_{i-1}) = p(x_i|x_{i-1})$$

That is,

$$p(x_1 \ldots x_m) = \prod_{i=1}^{m} p(x_i|x_{i-1})$$

where $x_0 = *$ is a special token at the start of every sentence.

Number of parameters: $O(|V|^2)$

Trigram Language Model

Assumption. A word is independent of all previous words conditioning on the two preceding words:

$$p(x_i|x_1 \dots x_{i-1}) = p(x_i|x_{i-2}, x_{i-1})$$

That is,

$$p(x_1 \dots x_m) = \prod_{i=1}^{m} p(x_i|x_{i-2}, x_{i-1})$$

where $x_{-1}, x_0 = *$ are special tokens at the start of every sentence.

Number of parameters: $O(|V|^3)$

n-Gram Language Model

Assumption. A word is independent of all previous words conditioning on the $n-1$ preceding words:

$$p(x_i|x_1 \dots x_{i-1}) = p(x_i|x_{i-n+1}, \dots, x_{i-1})$$

Number of parameters: $O(|V|^n)$

This kind of conditional independence assumption ("depends only on the last $n-1$ states...") is called a **Markov assumption**.

- Is this a reasonable assumption for language modeling?

Overview

Probability of a Sentence

n-Gram Language Models

- Unigram, Bigram, Trigram Models
- Estimation from Data
- Evaluation
- Smoothing

Log-Linear Language Models

A Practical Question

- Summary so far: We have designed **probabilistic language models** parametrized by finitely many values.

- Bigram model: Stores a **table** of $O(|V|^2)$ values

  $$q(x'|x) \qquad\qquad \forall x, x' \in V$$

  (plus $q(x|*)$ and $q(\texttt{STOP}|x)$) representing **transition probabilities** and computes

  $$\begin{aligned} p(\texttt{the cat barked}) = & q(\texttt{the}|*)\times \\ & q(\texttt{cat}|\texttt{the})\times \\ & q(\texttt{barked}|\texttt{cat}) \\ & q(\texttt{STOP}|\texttt{barked}) \end{aligned}$$

- **Q.** But where do we get these values?

Estimation from Data

- Our data is a **corpus** of N sentences $x^{(1)} \dots x^{(N)}$.

- Define **count**(x, x') to be the number of times x, x' appear together (called "bigram counts"):

$$\textbf{count}(x, x') = \sum_{i=1}^{N} \sum_{\substack{j=1: \\ x_j = x' \\ x_{j-1} = x}}^{l_i+1} 1$$

(l_i = length of $x^{(i)}$ and $x_{l_i+1} = \texttt{STOP}$)

- Define **count**$(x) := \sum_{x'}$ **count**(x, x') (called "unigram counts").

Example Counts

Corpus:

- `the dog chased the cat`
- `the cat chased the mouse`
- `the mouse chased the dog`

Example bigram/unigram counts:

$$\textbf{count}(x_0, \texttt{the}) = 3 \qquad \textbf{count}(\texttt{the}) = 6$$
$$\textbf{count}(\texttt{chased}, \texttt{the}) = 3 \qquad \textbf{count}(\texttt{chased}) = 3$$
$$\textbf{count}(\texttt{the}, \texttt{dog}) = 2 \qquad \textbf{count}(x_0) = 3$$
$$\textbf{count}(\texttt{cat}, \texttt{STOP}) = 1 \qquad \textbf{count}(\texttt{cat}) = 2$$

Parameter Estimates

- For all x, x' with **count**$(x, x') > 0$, set

$$q(x'|x) = \frac{\textbf{count}(x, x')}{\textbf{count}(x)}$$

  Otherwise $q(x'|x) = 0$.

- In the previous example:

$$\begin{aligned} q(\texttt{the}|x_0) &= 3/3 = 1 \\ q(\texttt{chased}|\texttt{dog}) &= 1/3 = 0.\bar{3} \\ q(\texttt{dog}|\texttt{the}) &= 2/6 = 0.\bar{3} \\ q(\texttt{STOP}|\texttt{cat}) &= 1/2 = 0.5 \\ q(\texttt{dog}|\texttt{cat}) &= 0 \end{aligned}$$

- Called **maximum likelihood estimation (MLE)**.

Justification of MLE

Claim. The solution of the constrained optimization problem

$$q^* = \underset{\substack{q:\ q(x'|x) \geq 0\ \forall x,x' \\ \sum_{x' \in V} q(x'|x) = 1 \forall x}}{\arg\max} \sum_{i=1}^{N} \sum_{j=1}^{l_i+1} \log q(x_j|x_{j-1})$$

is given by

$$q^*(x'|x) = \frac{\textbf{count}(x, x')}{\textbf{count}(x)}$$

(Proof?)

MLE: Other n-Gram Models

Unigram:

$$q(x) = \frac{\textbf{count}(x)}{N}$$

Bigram:

$$q(x'|x) = \frac{\textbf{count}(x, x')}{\textbf{count}(x)}$$

Trigram:

$$q(x''|x, x') = \frac{\textbf{count}(x, x', x'')}{\textbf{count}(x, x')}$$

Overview

Probability of a Sentence

n-Gram Language Models

- Unigram, Bigram, Trigram Models
- Estimation from Data
- Evaluation
- Smoothing

Log-Linear Language Models

Evaluation of a Language Model

"How good is the model at predicting **unseen** sentences?"

Held-out corpus:

Used for <u>evaluation purposes only</u>

Do not use held-out data for training the model!

Popular evaluation metric: **perplexity**

What We Are Doing: Conditional Density Estimation

- True context-word pairs distributed as $(c, w) \sim p_{CW}$
- We define some language model $q_{W|C}$
- Learning: minimize **cross entropy** between $p_{W|C}$ and $q_{W|C}$

$$H(p_{W|C}, q_{W|C}) = \mathop{\mathbf{E}}_{(c,w)\sim p_{CW}} \left[-\ln q_{W|C}(w|c) \right]$$

  Number of nats to encode the behavior of $p_{W|C}$ using $q_{W|C}$

$$H(p_{W|C}) \leq H(p_{W|C}, q_{W|C}) \leq \ln |V|$$

- Evaluation of $q_{W|C}$: check how small $H(p_{W|C}, q_{W|C})$ is!
  - If the model class of $q_{W|C}$ is universally expressive, an optimal model $q^*_{W|C}$ will satisfy $H(p_{W|C}, q^*_{W|C}) = H(p_{W|C})$ with $q^*_{W|C} = p_{W|C}$.

Perplexity

- Exponentiated cross entropy

$$\mathrm{PP}(p_{W|C}, q_{W|C}) = e^{H(p_{W|C}, q_{W|C})}$$

- Interpretation: effective vocabulary size

$$e^{H(p_{W|C})} \leq \mathrm{PP}(p_{W|C}, q_{W|C}) \leq |V|$$

- Empirical estimation: given $(c_1, w_1) \ldots (c_N, w_N) \sim p_{CW}$,

$$\widehat{\mathrm{PP}}(p_{W|C}, q_{W|C}) = e^{-\frac{1}{N} \sum_{i=1}^{N} \ln q_{W|C}(w_i|c_i)}$$

What is the empirical perplexity when $q_{W|C}(w_i|c_i) = 1$ for all i? When $q_{W|C}(w_i|c_i) = 1/|V|$ for all i?

Example Perplexity Values for n-Gram Models

- Using vocabulary size $|V| = 50,000$ (Goodman, 2001)
  - Unigram: 955, Bigram: 137, Trigram: 74
- Modern neural language models: probably $\ll 20$
- The big question: what is the minimum perplexity achievable with machines?

Overview

Probability of a Sentence

n-Gram Language Models

- Unigram, Bigram, Trigram Models
- Estimation from Data
- Evaluation
- Smoothing

Log-Linear Language Models

Smoothing: Additive

In practice, it's important to smooth estimation to avoid zero probabilities for unseen words:

$$q^{\alpha}(x'|x) = \frac{\#(x, x') + \alpha}{\#(x) + \alpha\,|V|}$$

Also called **Laplace smoothing**:
https://en.wikipedia.org/wiki/Additive_smoothing

Smoothing: Interpolation

With limited data, enforcing generalization by using less context also helps:

$$\begin{aligned} q^{\text{smoothed}}(x''|x,x') = &\lambda_1 q^{\alpha}(x''|x,x') + \\ &\lambda_2 q^{\alpha}(x''|x') + \\ &\lambda_3 q^{\alpha}(x'') \end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and $\lambda_i \geq 0$. Called **linear interpolation**.

Many Other Smoothing Techniques

- Kneser-Ney smoothing: Section 3.5 of
  `https://web.stanford.edu/~jurafsky/slp3/3.pdf`
- Good-Turing estimator: the "missing mass" problem
  - On the Convergence Rate of Good-Turing Estimators (McAllester and Schapire, 2001)

Final Aside: Tokenization

- Text: initially a *single* string

  "Call me Ishmael."

- Naive tokenization: by space

  [Call, me, Ishmael.]

- English-specific tokenization using rules or statistical model:

  [Call, me, Ishmael, .]

- Language-independent tokenization learned from data
  - Wordpiece: https://arxiv.org/pdf/1609.08144.pdf
  - Byte-Pair Encoding: https://arxiv.org/pdf/1508.07909.pdf
  - Sentencepiece: https://www.aclweb.org/anthology/D18-2012.pdf

Overview

Probability of a Sentence

n-Gram Language Models

- Unigram, Bigram, Trigram Models
- Estimation from Data
- Evaluation
- Smoothing

Log-Linear Language Models

Bashing n-Gram Models

- **Model parameters**: probabilities $q(x'|x)$
  - Training requires *constrained* optimization

$$q^* = \operatorname*{arg\,max}_{\substack{q:\ q(x'|x) \geq 0\ \forall x, x' \\ \sum_{x' \in V} q(x'|x) = 1 \forall x}} \sum_{i=1}^{N} \sum_{j=1}^{l_i+1} \log q(x_j | x_{j-1})$$

  Though easy to solve, not clear how to develp more complex functions

- Brittle: function of raw n-gram identities
  - Generalizing to unseen n-grams require explicit smoothing (cumbersome)

Feature Function

- Design a feature representation $\phi(x_1 \ldots x_n) \in \mathbb{R}^d$ of any n-gram $x_1 \ldots x_n \in V^n$
- For example,

$$\phi(\text{dog saw}) = (0, 0, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0, 0)$$

might be a vector in $\{0, 1\}^{|V|+|V|^2}$ indicating the presence of unigrams "dog" and "saw" and also the bigram "dog saw"

The Softmax Function

- Given any $v \in \mathbb{R}^D$, we define $\mathrm{softmax}(v) \in [0,1]^D$ to be a vector such that

$$\mathrm{softmax}_i(v) = \frac{e^{v_i}}{\sum_{j=1}^{D} e^{v_j}} \qquad \forall i = 1 \ldots D$$

- Check nonnegativity and normalization
- Softmax transforms any length-D vector into a distribution over D items

Log-Linear (n-Gram) Language Models

- **Model parameter**: $w \in \mathbb{R}^d$

- Given $x_1 \dots x_{n-1}$, defines a conditional distribution over V by

$$q(x|x_1 \dots x_{n-1}; w) = \text{softmax}_x([w^\top \phi(x_1 \dots x_{n-1}, x)]_{x \in V})$$

- Reason it's called log-linear:

$$\ln q(x|x_1 \dots x_n; w) = w^\top \phi(x_1 \dots x_{n-1}, x) - \ln \sum_{x' \in V} e^{w^\top \phi(x_1 \dots x_{n-1}, x')}$$

Training: Unconstrained Optimization

- Assume N samples $(x_1^{(i)} \ldots x_n^{(i)}, x^{(i)})$, find

$$w^* = \underset{w \in \mathbb{R}^{|V| \times d}}{\arg\max} \underbrace{\sum_{i=1}^{N} \ln q(x^{(i)} | x_1^{(i)} \ldots x_n^{(i)}; w)}_{J(w)}$$

- Unlike n-gram models, there is no closed-form solution for $\max_w J(w)$

- But actually this optimization problem is more "standard" because we can just do gradient ascent
  - It can be checked that $J(w)$ is concave, so doing gradient ascent will get us W^*