COMS 4705.H: Language Models

Karl Stratos

January 27, 2017

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 $p(x_1 \ldots x_m)$ over all sentences $x_1 \ldots x_m$.

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

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

Overview

The Probability of a Sentence

The n-Gram Language Models

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

Smoothing, Discount Methods

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

$$p(x_1 \ldots x_m) \geq 0 \qquad \forall x_1 \ldots x_m \in \Omega$$

$$\sum_{x_1 \ldots x_m \in \Omega} p(x_1 \ldots x_m) = 1$$

- 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

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

  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?)

Challenge 2: Infinitely Many Distributions

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

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

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

Overview

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:

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

That is,

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

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: $O(|V|^3)$

The n-Gram Language Model

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

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

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

The Probability of a Sentence

The n-Gram Language Models

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

Smoothing, Discount Methods

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 $x^{(1)} \ldots x^{(N)}$.

- 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

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

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

Evaluation of a Language Model

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

Held-out corpus: Used for evaluation purposes only

- One metric: log likelihood of unseen sentences $y^{(1)} \dots y^{(T)}$

$$\mathsf{LL} = \sum_{i=1}^{T} \log p(y^{(i)})$$

- More popular metric: **perplexity** of the model:

$$\mathsf{PP} = 2^{-\frac{1}{M} \sum_{i=1}^{T} \log p(y^{(i)})}$$

where M is the number of words + `STOP` symbols

Motivation of Perplexity: The Branching Factor

- How many times do we expect to flip a coin until we get a head, if it comes up head with probability ϵ?
- $1/\epsilon$ times
  - Mean of the geometric distribution with parameter ϵ
- Examples
  - $\epsilon = 0.5$: expect to flip two times
  - $\epsilon = 0.1$: expect to flip ten times
  - $\epsilon = 0.001$: expect to flip a thousand times
- The higher the "branching factor" $1/\epsilon$ is, the more "surprised" the model.

The Branching Factor of Language Models

- For simplicity, assume a single sentence $y = y_1 \dots y_{M-1}$ STOP.
- The branching factor of the model at word y_i:

$$\frac{1}{p(y_i|y_1 \dots y_{i-1})}$$

- Geometric average of the branching factors:

$$\prod_{i=1}^{M} \left(\frac{1}{p(y_i|y_1 \dots y_{i-1})} \right)^{\frac{1}{M}}$$

Perplexity = Average Branching Factor

$$\prod_{i=1}^{M} \left(\frac{1}{p(y_i | y_1 \ldots y_{i-1})} \right)^{\frac{1}{M}} = \prod_{i=1}^{M} 2^{\log\left(\frac{1}{p(y_i | y_1 \ldots y_{i-1})}\right)^{\frac{1}{M}}}$$

Perplexity = Average Branching Factor

$$\prod_{i=1}^{M} \left(\frac{1}{p(y_i|y_1 \ldots y_{i-1})} \right)^{\frac{1}{M}} = \prod_{i=1}^{M} 2^{\log\left(\frac{1}{p(y_i|y_1 \ldots y_{i-1})}\right)^{\frac{1}{M}}}$$

$$= \prod_{i=1}^{M} 2^{-\frac{1}{M}\log p(y_i|y_1 \ldots y_{i-1})}$$

Perplexity = Average Branching Factor

$$\prod_{i=1}^{M}\left(\frac{1}{p(y_i|y_1\ldots y_{i-1})}\right)^{\frac{1}{M}} = \prod_{i=1}^{M} 2^{\log\left(\frac{1}{p(y_i|y_1\ldots y_{i-1})}\right)^{\frac{1}{M}}}$$

$$= \prod_{i=1}^{M} 2^{-\frac{1}{M}\log p(y_i|y_1 \ldots y_{i-1})}$$

$$= 2^{-\frac{1}{M}\sum_{i=1}^{M}\log p(y_i|y_1 \ldots y_{i-1})}$$

Perplexity = Average Branching Factor

$$\begin{aligned}
\prod_{i=1}^{M}\left(\frac{1}{p(y_i|y_1\ldots y_{i-1})}\right)^{\frac{1}{M}} &= \prod_{i=1}^{M} 2^{\log\left(\frac{1}{p(y_i|y_1\ldots y_{i-1})}\right)^{\frac{1}{M}}} \\
&= \prod_{i=1}^{M} 2^{-\frac{1}{M}\log p(y_i|y_1\ldots y_{i-1})} \\
&= 2^{-\frac{1}{M}\sum_{i=1}^{M}\log p(y_i|y_1\ldots y_{i-1})} \\
&= 2^{-\frac{1}{M}\log\prod_{i=1}^{M} p(y_i|y_1\ldots y_{i-1})}
\end{aligned}$$

Perplexity = Average Branching Factor

$$
\begin{aligned}
\prod_{i=1}^{M} \left(\frac{1}{p(y_i|y_1 \ldots y_{i-1})} \right)^{\frac{1}{M}} &= \prod_{i=1}^{M} 2^{\log\left(\frac{1}{p(y_i|y_1 \ldots y_{i-1})}\right)^{\frac{1}{M}}} \\
&= \prod_{i=1}^{M} 2^{-\frac{1}{M}\log p(y_i|y_1 \ldots y_{i-1})} \\
&= 2^{-\frac{1}{M}\sum_{i=1}^{M}\log p(y_i|y_1 \ldots y_{i-1})} \\
&= 2^{-\frac{1}{M}\log\prod_{i=1}^{M} p(y_i|y_1 \ldots y_{i-1})} \\
&= 2^{-\frac{1}{M}\log p(y)}
\end{aligned}
$$

Perplexity = Average Branching Factor

$$\begin{aligned}
\prod_{i=1}^{M}\left(\frac{1}{p(y_i|y_1\ldots y_{i-1})}\right)^{\frac{1}{M}} &= \prod_{i=1}^{M} 2^{\log\left(\frac{1}{p(y_i|y_1\ldots y_{i-1})}\right)^{\frac{1}{M}}} \\
&= \prod_{i=1}^{M} 2^{-\frac{1}{M}\log p(y_i|y_1\ldots y_{i-1})} \\
&= 2^{-\frac{1}{M}\sum_{i=1}^{M}\log p(y_i|y_1\ldots y_{i-1})} \\
&= 2^{-\frac{1}{M}\log\prod_{i=1}^{M} p(y_i|y_1\ldots y_{i-1})} \\
&= 2^{-\frac{1}{M}\log p(y)} \\
&= \text{PP}
\end{aligned}$$

Example Perplexity Values

- If the model perfectly predicts test sentence,

$$\text{PP} = 2^{-\frac{1}{M}\log\prod_{i=1}^{M} p(y_i|y_1 \ldots y_{i-1})} = 2^{-\frac{1}{M}\log 1} = 1$$

- If the model predicts words uniformly at random,

$$\text{PP} = 2^{-\frac{1}{M}\sum_{i=1}^{M}\log p(y_i|y_1 \ldots y_{i-1})} = 2^{-\frac{1}{M}\sum_{i=1}^{M}\log 1/|V|} = |V|$$

- Empirical values for $|V| = 50,000$ (Goodman, 2001)
  - Unigram: 955, Bigram: 137, Trigram: 74

Overview

The Probability of a Sentence

The n-Gram Language Models

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

Smoothing, Discount Methods

Smoothing

In practice, it's important to **smooth** estimation of higher-order models:

$$q^{\text{smoothed}}(x''|x,x') = \lambda_1 q(x''|x,x') + \\ \lambda_2 q(x''|x') + \\ \lambda_3 q(x'')$$

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

Discount Methods

At test time, how do we handle words that were **unobserved in the training corpus**?

- Naively, we assign probability 0 to the entire held-out data!

A solution: "steal" some probability mass from observed words and allocate it for unobserved words.

Called **discount methods**. Will cover more details in video lectures / textbook.