Natural Language Processing
Probabilistic Language Models

Felipe Bravo-Marquez

March 26, 2024

Overview

- The language modeling problem
- Trigram models
- Evaluating language models: perplexity
- Estimation techniques:
    1. Linear interpolation
    2. Discounting methods
- This slides are based on the course material by Michael Collins: `http://www.cs.columbia.edu/~mcollins/cs4705-spring2019/slides/lmslides.pdf`

The Language Modeling Problem

- We have some (finite) vocabulary, say $\mathcal{V} = \{$the, a, man, telescope, Beckham, two, . . .$\}$
- We have an (infinite) set of strings, $\mathcal{V}^*$.
- For example:
    - the STOP
    - a STOP
    - the fan STOP
    - the fan saw Beckham STOP
    - the fan saw saw STOP
    - the fan saw Beckham play for Real Madrid STOP
- Where STOP is a special symbol indicating the end of a sentence.

The Language Modeling Problem (Continued)

- We have a training sample of example sentences in English.
- We need to ”learn” a probability distribution p.
- p is a function that satisfies:

$$\sum_{x \in V^*} p(x) = 1$$

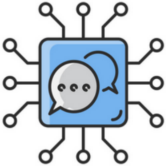
$$p(x) \geq 0 \quad \text{for all } x \in V^*$$

- Examples of probabilities assigned to sentences:

$$\begin{aligned}
p(\text{the STOP}) &= 10^{-12} \\
p(\text{the fan STOP}) &= 10^{-8} \\
p(\text{the fan saw Beckham STOP}) &= 2 \times 10^{-8} \\
p(\text{the fan saw saw STOP}) &= 10^{-15} \\
&\ldots \\
p(\text{the fan saw Beckham play for Real Madrid STOP}) &= 2 \times 10^{-9}
\end{aligned}$$

The Language Modeling Problem (Continued)

- Idea 1: The model assigns a higher probability to fluent sentences (those that make sense and are grammatically correct).
- Idea 2: Estimating this probability function from text (corpus).
- The language model helps text generation models distinguish between good and bad sentences.



Why would we want to do this?

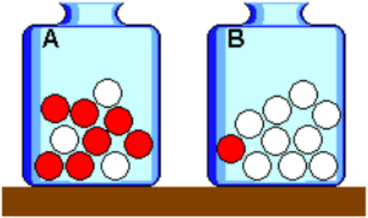
- Speech recognition was the original motivation.
- Consider the sentences: 1) recognize speech and 2) wreck a nice beach.
- These two sentences sound very similar when pronounced, making it challenging for automatic speech recognition systems to accurately transcribe them.
- When the speech recognition system analyzes the audio input and tries to transcribe it, it takes into account the language model probabilities to determine the most likely interpretation.
- The language model would favor p(recognize speech) over p(wreck a nice beach).
- This is because the former is a more common sentence and should occur more frequently in the training corpus.

Why on earth would we want to do this?

- By incorporating language models, speech recognition systems can improve accuracy by selecting the sentence that aligns better with linguistic patterns and context, even when faced with similar-sounding alternatives.
- Related problems are optical character recognition, handwriting recognition.
- Actually, Language Models are useful in any NLP tasks involving the generation of language (e.g., machine translation, summarization, chatbots).
- The estimation techniques developed for this problem will be VERY useful for other problems in NLP.

Language Models are Generative

- Language models can generate sentences by sequentially sampling from probabilities.
- This is analogous to drawing balls (words) from an urn where their sizes are proportional to their relative frequencies.
- Alternatively, one could always draw the most probable word, which is equivalent to predicting the next word.



A Naive Method

- A very naive method for estimating the probability of a sentence is to count the occurrences of the sentence in the training data and divide it by the total number of training sentences (N) to estimate the probability.
- We have N training sentences.
- For any sentence $x_1, x_2, \ldots, x_n$, $c(x_1, x_2, \ldots, x_n)$ is the number of times the sentence is seen in our training data.
- A naive estimate:

$$p(x1, x2, \ldots, xn) = \frac{c(x_1, x_2 \ldots, x_n)}{N}$$

- Problem: As the number of possible sentences grows exponentially with sentence length and vocabulary size, it becomes increasingly unlikely for a specific sentence to appear in the training data.
- Consequently, many sentences will have a probability of zero according to the naive model, leading to poor generalization.

Markov Processes

- Consider a sequence of random variables $X_1, X_2, \ldots, X_n$.
- Each random variable can take any value in a finite set V.
- For now, we assume the length n is fixed (e.g., $n = 100$).
- Our goal: model $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$

First-Order Markov Processes

$$
\begin{aligned}
P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) &= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \\
&= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_{i-1} = x_{i-1})
\end{aligned}
$$

The first-order Markov assumption: For any $i \in \{2, \ldots, n\}$ and any $x_1, \ldots, x_i$,

$$
P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) = P(X_i = x_i | X_{i-1} = x_{i-1})
$$

Second-Order Markov Processes

$$
\begin{aligned}
& P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \\
P(X_1 = x_1) \cdot P(X_2 = x_2 | X_1 = x_1) \cdot & \prod_{i=3}^{n} P(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1}) \\
& = \prod_{i=1}^{n} P(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})
\end{aligned}
$$

(For convenience, we assume $x_0 = x_{-1} = *$, where $*$ is a special ”start” symbol.)

Modeling Variable Length Sequences

- We would like the length of the sequence, n, to also be a random variable.
- A simple solution: always define $X_n = \text{STOP}$, where STOP is a special symbol.
- Then use a Markov process as before:

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

- (For convenience, we assume $x_0 = x_{-1} = *$, where $*$ is a special "start" symbol.)

Trigram Language Models

- A trigram language model consists of:
  1. A finite set $\mathcal{V}$
  2. A parameter $q(w|u, v)$ for each trigram u, v, w such that $w \in \mathcal{V} \cup \{\text{STOP}\}$, and $u, v \in \mathcal{V} \cup \{*\}$
- For any sentence $x_1 \ldots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \ldots (n-1)$, and $x_n = \text{STOP}$, the probability of the sentence under the trigram language model is:

$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-2}, x_{i-1})$$

- We define $x_0 = x_{-1} = *$ for convenience.

An Example

For the sentence `the dog barks STOP`, we would have:

$$p(\text{the dog barks STOP}) = q(\text{the}|*, *) \times q(\text{dog}|*, \text{the}) \times q(\text{barks}|\text{the, dog}) \times q(\text{STOP}|\text{dog, barks})$$

The Trigram Estimation Problem

Remaining estimation problem:

$$q(w_i | w_{i-2}, w_{i-1})$$

For example:

$$q(\text{laughs}|\text{the, dog})$$

A natural estimate (the ”maximum likelihood estimate”):

$$q(w_i | w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

For instance,

$$q(\text{laughs}|\text{the, dog}) = \frac{\text{Count(the, dog, laughs)}}{\text{Count(the, dog)}}$$

Sparse Data Problems

A natural estimate (the ”maximum likelihood estimate”):

$$q(w_i | w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

$$q(\text{laughs}|\text{the, dog}) = \frac{\text{Count(the, dog, laughs)}}{\text{Count(the, dog)}}$$

- Say our vocabulary size is $N = |V|$, then there are N^3 parameters in the model.
- For example, $N = 20,000 \Rightarrow 20,000^3 = 8 \times 10^{12}$ parameters.

Evaluating a Language Model: Perplexity

- We have some test data, m sentences: $s_1, s_2, s_3, \ldots, s_m$
- We could look at the probability under our model $\prod_{i=1}^{m} p(s_i)$. Or more conveniently, the log probability:

$$\log\left(\prod_{i=1}^{m} p(s_i)\right) = \sum_{i=1}^{m} \log p(s_i)$$

- In fact, the usual evaluation measure is perplexity:

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \frac{1}{M}\sum_{i=1}^{m} \log p(s_i)$$

- M is the total number of words in the test data

Some Intuition about Perplexity

- Say we have a vocabulary V, and $N = |V| + 1$, and a model that predicts:

$$q(w|u,v) = \frac{1}{N} \quad \text{for all } w \in V \cup \{\text{STOP}\}, \text{ for all } u,v \in V \cup \{*\}$$

- It's easy to calculate the perplexity in this case:

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \log \frac{1}{N} \Rightarrow \text{Perplexity} = N$$

- Perplexity can be seen as a measure of the effective "branching factor"

Some Intuition about Perplexity

- **Proof:** Let's asume we have m sentences of length n in the corpus, and M the amount of tokens in the corpus, $M = m \cdot n$.
- Let's consider the log (base 2) probability of a sentence $s = w_1 w_2 \dots w_n$ under the model:

$$\log p(s) = \log \prod_{i=1}^{n} q(w_i | w_{i-2}, w_{i-1}) = \sum_{i=1}^{n} \log q(w_i | w_{i-2}, w_{i-1})$$

- Since each $q(w_i | w_{i-2}, w_{i-1})$ is equal to $\frac{1}{N}$, we have:

$$\log p(s) = \sum_{i=1}^{n} \log \frac{1}{N} = n \cdot \log \frac{1}{N} = -n \cdot \log N$$

$$l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i) = \frac{1}{M} \sum_{i=1}^{m} -n \cdot \log N = \frac{1}{M} \cdot -m \cdot n \cdot \log N = -\log N$$

- Therefore, the perplexity is given by:

$$\text{Perplexity} = 2^{-l} = 2^{-(-\log N)} = N$$

Typical Values of Perplexity

- Results from Goodman ("A bit of progress in language modeling"), where $|V| = 50,000$ [Goodman, 2001].
- A trigram model: $p(x_1, \ldots, x_n) = \prod_{i=1}^{n} q(x_i|x_{i-2}, x_{i-1})$
  Perplexity = 74
- A bigram model: $p(x_1, \ldots, x_n) = \prod_{i=1}^{n} q(x_i|x_{i-1})$
  Perplexity = 137
- A unigram model: $p(x_1, \ldots, x_n) = \prod_{i=1}^{n} q(x_i)$
  Perplexity = 955

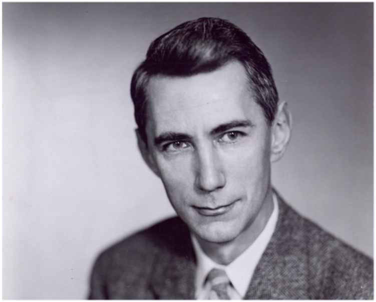
Some History

- Shannon conducted experiments on the entropy of English, specifically investigating how well people perform in the perplexity game.
- Reference: C. Shannon. “Prediction and entropy of printed English.” *Bell Systems Technical Journal*, 30:50–64, 1951. [Shannon, 1951]

Prediction and Entropy of Printed English

By C. E. SHANNON

(*Manuscript Received Sept. 15, 1950*)

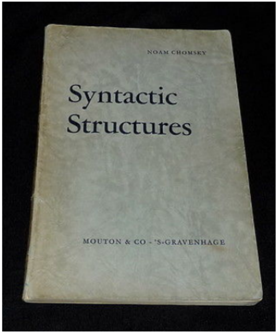


Some History

- Chomsky, in his book *Syntactic Structures* (1957), made several important points regarding grammar. [Chomsky, 2009]
- According to Chomsky, the notion of ”grammatical” cannot be equated with ”meaningful” or ”significant” in a semantic sense.
- He illustrated this with two nonsensical sentences:
    - (1) Colorless green ideas sleep furiously.
    - (2) Furiously sleep ideas green colorless.
- While both sentences lack meaning, Chomsky argued that only the first one is considered grammatical by English speakers.

Some History

- Chomsky also emphasized that grammaticality in English cannot be determined solely based on statistical approximations.
- Even though neither sentence (1) nor (2) has likely occurred in English discourse, a statistical model would consider them equally ”remote” from English.
- However, sentence (1) is grammatical, while sentence (2) is not, highlighting the limitations of statistical approaches in capturing grammaticality.



The Bias-Variance Trade-Off

- Trigram maximum-likelihood estimate:

$$q_{\mathrm{ML}}(w_i|w_{i-2}, w_{i-1}) = \frac{\mathrm{Count}(w_{i-2}, w_{i-1}, w_i)}{\mathrm{Count}(w_{i-2}, w_{i-1})}$$

- Bigram maximum-likelihood estimate:

$$q_{\mathrm{ML}}(w_i|w_{i-1}) = \frac{\mathrm{Count}(w_{i-1}, w_i)}{\mathrm{Count}(w_{i-1})}$$

- Unigram maximum-likelihood estimate:

$$q_{\mathrm{ML}}(w_i) = \frac{\mathrm{Count}(w_i)}{\mathrm{Count}()}$$

Linear Interpolation

- Take our estimate $q(w_i|w_{i-2}, w_{i-1})$ to be

$$q(w_i|w_{i-2}, w_{i-1}) = \lambda_1 \cdot q_{\mathrm{ML}}(w_i|w_{i-2}, w_{i-1}) + \lambda_2 \cdot q_{\mathrm{ML}}(w_i|w_{i-1}) + \lambda_3 \cdot q_{\mathrm{ML}}(w_i)$$

  where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i.
- Our estimate correctly defines a distribution (define $V' = V \cup \{\mathrm{STOP}\}$):

$$\begin{aligned}
& \sum_{w \in V'} q(w|u, v) \\
= & \sum_{w \in V'} [\lambda_1 \cdot q_{\mathrm{ML}}(w|u, v) + \lambda_2 \cdot q_{\mathrm{ML}}(w|v) + \lambda_3 \cdot q_{\mathrm{ML}}(w)] \\
= & \lambda_1 \sum_w q_{\mathrm{ML}}(w|u, v) + \lambda_2 \sum_w q_{\mathrm{ML}}(w|v) + \lambda_3 \sum_w q_{\mathrm{ML}}(w) \\
= & \lambda_1 + \lambda_2 + \lambda_3 = 1
\end{aligned}$$

- We can also show that $q(w|u, v) \geq 0$ for all $w \in V'$.

Estimating λ Values

- Hold out part of the training set as *validation* data.
- Define $c'(w_1, w_2, w_3)$ to be the number of times the trigram (w_1, w_2, w_3) is seen in the validation set.
- Choose λ_1, λ_2, λ_3 to maximize:

$$L(\lambda_1, \lambda_2, \lambda_3) = \sum_{w_1, w_2, w_3} c'(w_1, w_2, w_3) \log q(w_3 | w_1, w_2)$$

such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i, and where

$$q(w_i | w_{i-2}, w_{i-1}) = \lambda_1 \cdot q_{\text{ML}}(w_i | w_{i-2}, w_{i-1}) + \lambda_2 \cdot q_{\text{ML}}(w_i | w_{i-1}) + \lambda_3 \cdot q_{\text{ML}}(w_i)$$

Discounting Methods

- Consider the following counts and maximum-likelihood estimates:

Sentence	Count	$q_{\mathbf{ML}}(w_i \mid w_{i-1})$
the	48	
the, dog	15	$15/48$
the, woman	11	$11/48$
the, man	10	$10/48$
the, park	5	$5/48$
the, job	2	$2/48$
the, telescope	1	$1/48$
the, manual	1	$1/48$
the, afternoon	1	$1/48$
the, country	1	$1/48$
the, street	1	$1/48$

- The maximum-likelihood estimates are high, particularly for low count items.

Discounting Methods

- Define "discounted" counts as follows:

$$\text{Count}^*(x) = \text{Count}(x) - 0.5$$

Sentence	Count	$\mathbf{Count^*(x)}$	$\mathbf{q_{ML}(w_i \mid w_{i-1})}$
the	48		
the, dog	15	14.5	$14.5/48$
the, woman	11	10.5	$10.5/48$
the, man	10	9.5	$9.5/48$
the, park	5	4.5	$4.5/48$
the, job	2	1.5	$1.5/48$
the, telescope	1	0.5	$0.5/48$
the, manual	1	0.5	$0.5/48$
the, afternoon	1	0.5	$0.5/48$
the, country	1	0.5	$0.5/48$
the, street	1	0.5	$0.5/48$

- The new estimates are based on the discounted counts.

Discounting Methods (Continued)

- We now have some ”missing probability mass”:

$$\alpha(w_{i-1}) = 1 - \sum_{w} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

- For example, in our case:

$$\alpha(\text{the}) = \frac{10 \times 0.5}{48} = \frac{5}{48}$$

Katz Back-Off Models (Bigrams)

- For a bigram model, define two sets:

$$A(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) > 0\}$$

$$B(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) = 0\}$$

- A bigram model:

$$q_{\text{BO}}(w_i|w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-1}, w_i)}{\text{Count}(w_{i-1})} & \text{if } w_i \in A(w_{i-1}) \\ \frac{\alpha(w_{i-1}) q_{\text{ML}}(w_i)}{\sum_{w \in B(w_{i-1})} q_{\text{ML}}(w)} & \text{if } w_i \in B(w_{i-1}) \end{cases}$$

- Where:

$$\alpha(w_{i-1}) = 1 - \sum_{w \in A(w_{i-1})} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

Katz Back-Off Models (Trigrams)

- For a trigram model, first define two sets:

$$A(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0\}$$

$$B(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) = 0\}$$

- A trigram model is defined in terms of the bigram model:

$$q_{\text{BO}}(w_i | w_{i-2}, w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} & \text{if } w_i \in A(w_{i-2}, w_{i-1}) \\ \frac{\alpha(w_{i-2}, w_{i-1}) q_{\text{BO}}(w_i | w_{i-1})}{\sum_{w \in B(w_{i-2}, w_{i-1})} q_{\text{BO}}(w | w_{i-1})} & \text{if } w_i \in B(w_{i-2}, w_{i-1}) \end{cases}$$

- Where:

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in A(w_{i-2}, w_{i-1})} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w)}{\text{Count}(w_{i-2}, w_{i-1})}$$

Summary

- Deriving probabilities in probabilistic language models involves three steps:
    1. Expand $p(w_1, w_2, \ldots, w_n)$ using the Chain rule.
    2. Apply Markov Independence Assumptions $p(w_i|w_1, w_2, \ldots, w_{i-2}, w_{i-1}) = p(w_i|w_{i-2}, w_{i-1})$.
    3. Smooth the estimates using low order counts.
- Other methods for improving language models include:
    - Introducing latent variables to represent topics, known as topic models. [Blei et al., 2003]
    - Replacing $p(w_i|w_1, w_2, \ldots, w_{i-2}, w_{i-1})$ with a predictive neural network and an "embedding layer" to better represent larger contexts and leverage similarities between words in the context. [Bengio et al., 2000]
- Modern language models utilize deep neural networks in their backbone and have a vast parameter space.

Questions?

Thanks for your Attention!

References I

Bengio, Y., Ducharme, R., and Vincent, P. (2000).
A neural probabilistic language model.
Advances in neural information processing systems, 13.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003).
Latent dirichlet allocation.
Journal of machine Learning research, 3(Jan):993–1022.

Chomsky, N. (2009).
Syntactic structures.
In *Syntactic Structures*. De Gruyter Mouton.

Goodman, J. T. (2001).
A bit of progress in language modeling.
Computer Speech & Language, 15(4):403–434.

Shannon, C. E. (1951).
Prediction and entropy of printed english.
Bell system technical journal, 30(1):50–64.