N-Gram Language Models

ChengXiang Zhai

Department of Computer Science

University of Illinois, Urbana-Champaign

The Data and Information Systems Laboratories
at The University of Illinois at Urbana-Champaign
Large Scale Information Management and Mining

Outline

- General questions to ask about a language model
- N-gram language models
- Special case: Unigram language models
- Smoothing Methods

Central Questions to Ask about a LM: "ADMI"

- **Application**: Why do you need a LM? For what purpose?

  → Evaluation metric for a LM

  Speech recognition
  Perplexity

- **Data:** What kind of data do you want to model?

  → Data set for estimation & evaluation

  Speech text data

- **Model:** How do you define the model?

  → Assumptions to be made

  Limited memory
  N-gram LM

- **Inference:** How do you infer/estimate the parameters?

  → Inference/Estimation algorithm

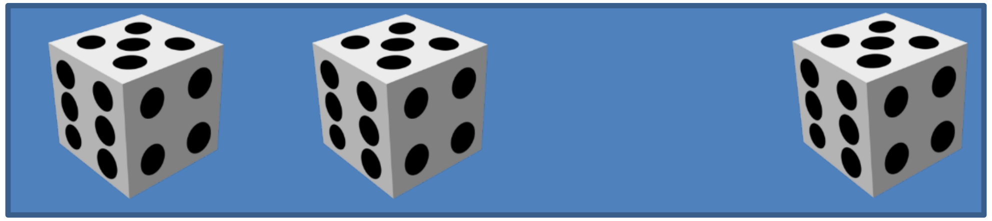
  Smoothing methods

Central Question in LM: $p(w_1w_2\ldots w_m|C)$=?

- What is C? We usually ignore C (="context") since it depends on the application, but it's important to consider it when applying a LM
  - Refinement 1: $p(w_1w_2\ldots w_m|C) \approx p(w_1w_2\ldots w_m)$
- What random variables are involved? What is the event space?
  - What event does "$w_1w_2\ldots w_m$" represent? What is the sample space?
  - $p(w_1w_2\ldots w_m)$ =$p(X=w_1w_2\ldots w_m)$? vs. $p(X_1=w_1, X_2=w_2,\ldots X_m=w_m)$?

X=?

X_1=? X_2=? X_m=?



Central Question in LM: $p(w_1 w_2 \ldots w_m | C)$=?

- Refinement 2: $p(w_1 w_2 \ldots w_m) = p(X_1=w_1, X_2=w_2, \ldots X_m=w_m)$
  - What assumption have we made here?
- Chaining Rule: $p(w_1 w_2 \ldots w_m) = p(w_1)p(w_2|w_1) \ldots p(w_m|w_1 w_2 \ldots w_{m-1})$
  - What about $p(w_1 w_2 \ldots w_m) = p(w_m)p(w_{m-1}|w_m) \ldots p(w_1|w_2 \ldots w_m)$?
- Refinement 3: Assume limited dependence (only depends on

$$p(X_1=w_1, X_2=w_2, \ldots X_m=w_m) \approx p(X_1=w_1)\ p(X_2=w_2|X_1=w_1) \ldots p(X_n=w_n|X_1=w_1, \ldots, X_{n-1}=w_{n-1}) \ldots$$

$$\mathbf{p(w_1 w_2 \ldots w_m) \approx p(w_1)\ p(w_2|w_1) \ldots p(w_n|w_1, \ldots, w_{n-1}) \ldots\ p(w_m|w_{m-n+1}, \ldots, w_{m-1})}$$

Key Assumption in N-gram LM:

$$p(w_m | \boxed{w_1,\ldots,}\ w_{m-n+1}\ldots,w_{m-1}) = p(w_m | w_{m-n+1},\ldots,w_{m-1})$$

Ignored

Does this assumption hold?

Estimation of N-Gram LMs

- Text Data: D
- Question: $p(w_m | w_{m-n+1},\ldots,w_{m-1})=?$

$P(X|Y)=p(X,Y)/p(Y)$

$$p(w_m | w_{m-n+1},\ldots,w_{m-1}) = \frac{p(w_{m-n+1},\ldots,w_{m-1},w_m)}{p(w_{m-n+1},\ldots,w_{m-1})}$$

- Boils down to estimate $p(w_1,w_2,\ldots,w_m)$, ML estimate is:

$$p(w_1,w_2,\ldots,w_m) = \frac{c(w_1 w_2 \ldots w_m, D)}{\sum_{u_i \in V} c(u_1 u_2 \ldots u_m, D)}$$

Count of word sequence "$w_1 w_2 \ldots w_m$"

Total counts of all word sequences of length m

ML Estimate of N-Gram LM

$$p(w_m | w_{m-n+1}, \ldots, w_{m-1}) = \frac{c(w_{m-n+1} \ldots w_{m-1} w_m, D)}{\sum_{u \in V} c(w_{m-n+1} \ldots w_{m-1} u, D)}$$

- Count of long word sequences may be zero!
  - Not accurate
  - Cause problems when computing the conditional probability $p(w_m | w_{m-n+1}, \ldots, w_{m-1})$
- Solution: smoothing
  - **Key idea: backoff to shorter N-grams, eventually to unigrams**
  - **Treat shorter N-gram models as prior in Bayesian estimation**

Special Case of N-Gram LM: Unigram LM

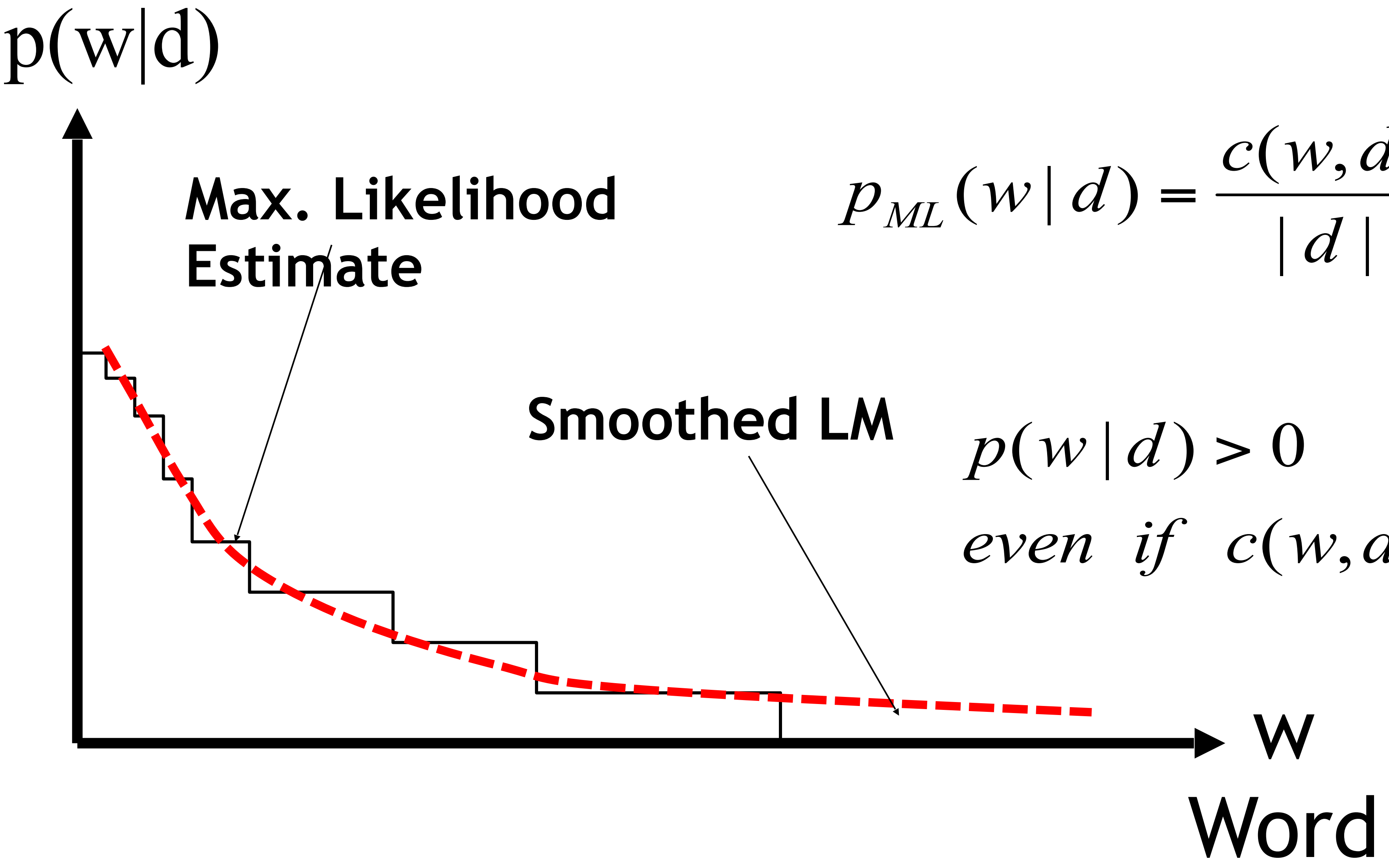
- Generate text by generating each word INDEPENDENTLY
- **$p(w_m|w_1,\ldots,w_{m-n+1}\ldots,w_{m-1})=p(w_m)$: History didn't matter!**
- How to estimate a unigram LM?
  - Text data: d
  - Maximum Likelihood estimator:

$$p_{ML}(w|d)$$

Count of word w in d

Total Counts of all words in d

Unigram Language Model Smoothing (Illustration)



$$p_{ML}(w|d) = \frac{c(w,d)}{|d|}$$

$$p(w|d) > 0$$
$$even\ \ if\ \ c(w,d) = 0$$

How to Smooth?

- All smoothing methods try to
  - discount the probability of words seen in a text data set
  - re-allocate the extra counts so that unseen words will have a non-zero count

- Method 1: Additive smoothing: **Add a constant δ to the counts of each word, e.g., "add 1"**

Count of w in d

$$p(w|d) = \frac{c(w,d)+1}{|d|+|V|}$$

"Add one", Laplace

Vocabulary size

Length of d (total counts)

Improve Additive Smoothing

- Should all unseen words get equal probabilities?
- We can use a reference model to discriminate unseen words

Discounted ML estimate

$$p(w|d) = \begin{cases} p_{DML}(w|d) & \text{if } w \text{ is seen in } d \\ \alpha_d \, p(w|REF) & \text{otherwise} \end{cases}$$

Reference language model

$$\alpha_d = \frac{1 - \sum_{w \text{ is seen}} p_{DML}(w|d)}{\sum_{w \text{ is unseen}} p(w|REF)}$$

Normalizer
Prob. Mass for unseen words

However, how do we define p(w|REF)?

- p(w|REF): Reference Language Model
- What do we know about those unseen words?
- Why are there unseen words?
  - Zipf's law: most words occur infrequently in text (e.g., just once)
  - Unseen words are non-relevant to a topic
  - Unseen words are relevant, but the text data sample isn't large enough to include them
- The context variable C in $p(w_1 w_2 \ldots w_m | C)$ can provide a basis for defining p(w|REF)
  - E.g., in retrieval, p(w|Collection) can serve as p(w|REF) for estimating a language model for an individual document p(w|d)

Interpolation vs. Backoff

- **Interpolation:** view p(w|REF) as a prior and the actual counts as observed evidence

$$p(w|d) = (1-\lambda)\frac{c(w,d)}{|d|} + \lambda\, p(w|REF)$$

- **Backoff** (Katz-Backoff): if the count is sufficiently high (sufficient evidence), we'd trust the ML estimate, otherwise, we simply ignore the ML estimate and go for p(w|REF)

$$p(w|d) = \begin{cases} \beta\frac{c(w,d)}{|d|} & if\ \ c(w,d) > k \\ \lambda p(w|REF) & otherwise \end{cases}$$

Smoothing Methods based on Interpolation

- Method 2: Absolute discounting (Kneser-Ney Smoothing): **Subtract a constant δ from the count of each word**

 # unique words

$$p(w|d) = \frac{\max(c(w,d)-\delta,0)+\delta |d|_u \, p(w|REF)}{|d|}$$

- Method 3: Linear interpolation (Jelinek-Mercer smoothing): **"Shrink" uniformly toward p(w|REF)**

$$p(w|d) = (1-\lambda)\boxed{\frac{c(w,d)}{|d|}} + \lambda \, p(w|REF)$$

ML estimate **parameter**

Smoothing Methods based on Interpolation (cont.)

- Method 4 Dirichlet Prior/Bayesian (McKay): **Assume pseudo counts $\mu p(w|REF)$**

parameter

$$p(w|d) = \frac{c(w,d) + \mu\, p(w|REF)}{|d| + \mu} = \frac{|d|}{|d| + \mu} \frac{c(w,d)}{|d|} + \frac{\overset{\text{parameter}}{\mu}}{|d| + \mu} p(w|REF)$$

What would happen if we increase/decrease μ? What if |d| ➔ $+\infty$

Linear Interpolation (Jelinek-Mercer) Smoothing

Unigram LM p(w|θ)=?

	…
10/100	**text ?**
5/100	mining ?
3/100	association ?
3/100	database ?
	…
1/100	query ?
0/100	**network?**

Document d

Total #words=**100**

text 10
mining 5
association 3
database 3
algorithm 2
…
query 1
efficient 1

Collection LM

P(w|C)

the 0.1
a 0.08
..
computer 0.02
database 0.01
……
text 0.001
network 0.001
mining 0.0009
…

$$p(w\,|\,d) = (1-\lambda)\frac{c(w,d)}{|d|} + \lambda\, p(w\,|\,C)$$

$$\lambda \in [0,1]$$

$$p(\text{"text"}\,|\,d) = (1-\lambda)\frac{10}{100} + \lambda * 0.001$$

$$p(\text{"network"}\,|\,d) = \lambda * 0.001$$

Dirichlet Prior (Bayesian) Smoothing

Unigram LM p(w|θ)=?

	…
10/100	**text ?**
5/100	mining ?
3/100	association ?
3/100	database ?
	…
1/100	query ?
0/100	**network?**

Document d
Total #words=**100**

text 10
mining 5
association 3
database 3
algorithm 2
…
query 1
efficient 1

Collection LM
P(w|C)

the 0.1
a 0.08
..
computer 0.02
database 0.01
……
text 0.001
network 0.001
mining 0.0009
…

$$p(w|d) = \frac{c(w;d) + \mu p(w|C)}{|d| + \mu} = \frac{|d|}{|d|+\mu}\frac{c(w,d)}{|d|} + \frac{\mu}{|d|+\mu} p(w|C)$$

μ

$$p(\text{"text"}|d) = \frac{10 + \mu * 0.001}{100 + \mu}$$

$$p(\text{"network"}|d) = \frac{\mu}{100 + \mu} * 0.001$$

Dirichlet Prior Smoothing (Bayesian Smoothing)

- **Bayesian estimator** of multinomial distribution (unigram LM)
  - First consider posterior of parameters: $p(\theta|d) \propto p(d|\theta)p(\theta)$
  - Then, consider the mean or mode of the posterior distribution
- **Sampling distribution**(of data): $p(d|\theta)$
- **Prior** (on model parameters): $P(\theta)=p(\theta_1, \ldots, \theta_N)$, where θ_1 is probability of the i-th word in the vocabulary
- **Conjugate Prior**: intuitive & mathematically convenient
  - "encode" the prior as **"extra pseudo counts,"** which can be **conveniently combined with the observed actual counts**
  - $p(d|\theta)$ and $p(\theta)$ have the **same functional form**

Dirichlet Prior Smoothing

- Dirichlet distribution is a conjugate prior for multinomial sampling distribution

"pseudo" word counts $\alpha_i = \mu\, p(w_i|REF)$

$$Dir(\theta \mid \alpha_1, \boxed{?}, \alpha_N) = \frac{\Gamma(\alpha_1 + \boxed{?} + \alpha_N)}{\Gamma(\alpha_1)\boxed{?}\Gamma(\alpha_N)} \prod_{i=1}^{N} \theta_i^{\alpha_i - 1}$$

X

$$p(d \mid \theta) = \frac{|d|!}{c(w_1)! \ldots c(w_N)!} \prod_{i=1}^{N} \theta_i^{c(w_i, d)}$$

$$p(\theta \mid d) = Dir(\theta \mid \alpha_1 + c(w_1), \boxed{?}, \alpha_N + c(w_N))$$

$$= \frac{\Gamma(\alpha_1 + \boxed{?} + \alpha_N + |d|)}{\Gamma(\alpha_1 + w_1)\boxed{?}\Gamma(\alpha_N + w_N)} \prod_{i=1}^{N} \theta_i^{c(w_i) + \alpha_i - 1}$$

Dirichlet Prior Smoothing (cont.)

Posterior distribution of parameters:

$$p(\theta \mid d) = Dir(\theta \mid c(w_1)+\alpha_1, \boxed{?}, c(w_N)+\alpha_N)$$

$$\text{Property: If } \theta \sim Dir(\theta \mid \alpha), \text{ then } E(\theta) = \{\frac{\alpha_i}{\sum \alpha_i}\}$$

The predictive distribution is the same as the mean:

$$\hat{\theta}_i = p(w_i \mid \hat{\theta}) = \int p(w_i \mid \theta) Dir(\theta \mid \alpha) d\theta$$

$$= \frac{c(w_i)+\alpha_i}{|d| + \sum_{i=1}^{N} \alpha_i} = \boxed{\frac{c(w_i)+\mu p(w_i \mid REF)}{|d|+\mu}}$$

Dirichlet prior smoothing

The Data and Information Systems Laboratories at The University of Illinois at Urbana-Champaign Large Scale Information Management and Mining

TIMAN

Good Turing Smoothing

- Key Idea: **Assume total # unseen events to be n_1 (# of singletons), and adjust all the seen events in the same way**

Adjusted count

Sum of counts of all terms that occurred c(w,d)+1 times

$$p(w|d) = \frac{c^*(w,d)}{|d|};\ \ c^*(w,d) = \frac{c(w,d)+1}{n_{c(w,d)}} n_{c(w,d)+1};\ 0^* = \frac{n_1}{n_0}, 1^* = \frac{2 * n_2}{n_1}, \ldots..$$

Share the counts among all the words that occurred c(w,d) times

$$n_r = the\ \ number\ \ of\ words\ \ with\ \ count\ \ r$$

$$What\ \ if\ \ n_{c(w,d)} = 0?\ \ What\ \ about\ \ p(w|REF)?$$

Heuristics are needed

Smoothing of $p(w_m | w_1, \ldots, w_{m-n+1} \ldots, w_{m-1})$

$$p(w_m | w_{m-n+1}, \ldots, w_{m-1}) = \frac{c(w_m, w_{m-n+1}, \ldots, w_{m-1}; D)}{\sum_{u \in V} c(u, w_{m-n+1}, \ldots, w_{m-1}; D)}$$

What if this is zero?

- How should we define p(w|REF)?
- In general, p(w|REF) can be defined based on any "clues" from the history **h=**$(w_{m-n+1}, \ldots, w_{m-1})$
  - **Most natural: p(w|REF)=**$p(w_m | w_{m-n+2}, \ldots, w_{m-1})$**, ignore** w_{m-n+1} **; can be done recursively to rely on shorter and shorter history**
  - **In general, relax the condition to make it less specific so as to increase the counts we can collect (e.g., shorten the history, cluster the history)**

What You Should Know

- What is an N-gram language model? What assumptions are made in an N-gram language model? What are the events involved?
- How to compute ML estimate of an N-gram language model?
- Why do we need to do smoothing in general?
- Know the major smoothing methods and how they work: additive smoothing, absolute discount, linear interpolation (fixed coefficient), Dirichlet prior, Good Turing
- Know the basic idea of deriving Dirichlet prior smoothing