

# Language Model (LM)

LING 570

Fei Xia

# LM

- Word prediction: to predict the next word in a sentence.
  - Ex: Once hw3 is\_\_\_\_, ....
  - Ex: The exam is next\_\_\_\_. His sabbatical is next\_\_\_\_.
- Statistical models of word sequences are called language models (LMs).
- Task:
  - Build a statistical model from the training data.
  - Now given a sentence  $w_1 w_2 \dots w_n$ , we want to estimate its probability  $P(w_1 \dots w_n)$ ?
- Goal: build a model that prefers good sentences to bad ones.

# Some Terms

- Corpus: a collection of text or speech
- Words: may or may not include punctuation marks.
- Word **types**: the number of distinct words in a corpus
- Word **tokens**: the total number of words in a corpus

# Applications of LMs

- Speech recognition:
  - Ex: I bought **two/too/to** books.
- Handwriting recognition:
- Machine translation:
  - Ex: (Chinese->English) I bought two **book/books**.
- Grammar checker:
- Language identification:
- ...

# LM task

- Training: given a corpus (i.e., training data), build a LM (e.g., learn probabilities of word sequences)
- Testing:
  - Given trained LM and new text, calculate the probability of the text
  - Given trained LM and multiple word sequences, select most probable sequence among those sequences.
- What does LM look like?
  - N-gram LM
  - Class-based LM
  - Structure LM
  - Neural LM

# Outline

- Motivation: LM applications
- N-gram LM
- Evaluation
- Other models and adaptation
- (Later) Neural LM

# N-gram LM

# N-gram LM

- Given a sentence  $w_1 w_2 \dots w_n$ , how to estimate  $P(w_1 \dots w_n)$ ?
- The Markov independence assumption:  
 $P(w_n \mid w_1, \dots, w_{n-1})$  depends only on the previous  $k$  words.
- $P(w_1 \dots w_n)$   
 $= P(w_1) * P(w_2 \mid w_1) * \dots P(w_n \mid w_1, \dots, w_{n-1})$   
 $\frac{1}{4} P(w_1) * P(w_2 \mid w_1) * \dots P(w_n \mid w_{n-k+1}, \dots, w_{n-1})$
- 0<sup>th</sup> order Markov model: unigram model
- 1<sup>st</sup> order Markov model: bigram model
- 2<sup>nd</sup> order Markov model: trigram model
- ...



# Unigram LM

- $P(w_1 \dots w_n)$   
 $\frac{1}{4} P(w_1) * P(w_2) * \dots * P(w_n)$
- Training stage: Estimating  $P(w)$ :
  - MLE:  $P(w) = C(w)/N$ ,  $N$  is the number of tokens
  - How many model parameters?
- Testing: calculate  $P(s)$  for a given sentence  $s$
- Can we represent an LM as a PFA?
  - what are the input symbols?
  - what are the probabilities on the arcs?
  - How many states in the PFA?

# Bigram LM

- $P(w_1 \dots w_n)$   
 $= P(\text{BOS } w_1 \dots w_n \text{ EOS})$   
 $\frac{1}{4} P(\text{BOS}) * P(w_1 | \text{BOS}) * \dots * P(w_n | w_{n-1}) * P(\text{EOS} | w_n)$
- Training stage: Estimating  $P(w_n | w_{n-1})$ :
  - MLE:  $P(w_n | w_{n-1}) = C(w_{n-1}, w_n) / C(w_{n-1})$
  - How many model parameters?
- Can we represent an LM as a PFA?
  - what are the input symbols?
  - what does a state represent?
  - what are the probabilities on the arcs?
  - How many states in the PFA?

# Trigram LM

- $P(w_1 \dots w_n) = P(\text{BOS } w_1 \dots w_n \text{ EOS})$   
 $\frac{1}{4}$   $P(\text{BOS}) * P(w_1 | \text{BOS}) * P(w_2 | \text{BOS}, w_1) * \dots$   
 $* P(w_n | w_{n-2}, w_{n-1}) * P(\text{EOS} | w_{n-1} w_n)$
- Training stage: Estimating  $P(w_n | w_{n-2}, w_{n-1})$ :
  - MLE:  $P(w_n | w_{n-2}, w_{n-1}) = C(w_{n-2}, w_{n-1}, w_n) / C(w_{n-2}, w_{n-1})$
  - How many model parameters?
- How many states in the PFA?

# Recap

- Ngrams:
  - # of FSA states:  $|V|^{n-1}$
  - # of model parameters:  $|V|^n$
- Issues:
  - Data sparseness, Out-of-vocabulary elements (OOV)
    - ➔ Smoothing
  - Mismatches between training & test data
  - Other Language Models

# Probabilistic Language Generation

- *Coin-flipping models*
  - A sentence is generated by a randomized algorithm
    - The generator can be in one of several “states”
    - Flip coins to choose the next state
    - Flip other coins to decide which letter or word to output

# LM built from Shakespeare's work

Unigram: To him swallowed confess hear both. Which. Of save  
on trail for are ay device and rote life have

Bigram: What means, sir. I confess she? then all sorts,  
he is trim, captain.

Trigram: Sweet prince, Falstaff shall die. Harry of  
Monmouth's grave.

4-gram: Will you not tell me who I am?  
It cannot be but so.

# LM built from the Wall Street Journal

*unigram:* Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

*bigram:* Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

*trigram:* They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

# N-gram LM packages

- SRI LM toolkit
- CMU LM toolkit
- ...



# Outline

- Motivation: LM applications
- N-gram LM
- **Evaluation**
- Other models and adaptation

# Evaluation (in general)

- Evaluation is required for almost all CompLing publications.
- Should be integrated early
- There are many factors to consider:
  - Data
  - Metrics
  - Results of competing systems
  - ...

# Evaluation guidelines

- Always evaluate your system
- Use standard metrics
- Separate training/dev/test data
- Use standard training/dev/test data
- Clearly specify experiment setting
- Include baseline and results from competing systems
- Perform error analysis
- Show the system is useful for real applications (optional)

# Division of data

- Training data
  - True training data: to learn model parameters
  - held-out data: to tune other parameters
- Development data: used when developing a system.
- Test data: used for the final, blind evaluation
- Dividing the data:
  - Common split: 80%, 10%, 10%.
  - N-fold cross validation

# Evaluating LMs

- Extrinsic evaluation (aka in vivo, or E2E)
  - Embed alternate models in system
  - See which improves overall application
    - ASR, MT, ...
- Intrinsic evaluation (or Unit test):
  - Metric applied directly to model
    - Independent of larger application
  - Ex: Perplexity
- Why not just extrinsic?

# Perplexity

- Perplexity is based on computing the probabilities of each sentence in the test set.
- Intuition:
  - A better model will have tighter fit to test data
  - It will yield higher probability on test data

# Perplexity

- Formally, with n-gram LM:

$$PPL(T) = P(w_1 \dots w_N)^{-\frac{1}{N}} = \frac{1}{\sqrt[N]{P(w_1 \dots w_N)}} = \frac{1}{\sqrt[N]{\prod_{i=1}^N P(w_i | w_1, \dots, w_{i-1})}}$$

- Ex: with trigram LM:

$$PPL(T) = \frac{1}{\sqrt[N]{\prod_{i=1}^N P(w_i | w_{i-2}, w_{i-1})}}$$

- Inversely related to probability of sequence
  - Higher probability → Lower perplexity

# Using log

$$x = a^{\log_a x}$$

$$\begin{aligned} PPL(T) &= P(T)^{-\frac{1}{N}} = 2^{\log_2 P(T) - \frac{1}{N}} \\ &= 2^{-\frac{1}{N} \log_2 P(T)} \\ &= 2^{H(L, P)} \end{aligned}$$

$$PPL(T) = 10^{-\frac{1}{N} \log_{10} P(T)} = 10^{-\frac{1}{N} \lg P(T)}$$



# Notations for logarithm

- binary log: the base is 2,  $\log_2(x)$ 
  - Ex: entropy
- natural log: the base is e,  $\ln(x)$ 
  - Ex: log linear model
- common log: the base is 10,  $\lg(x)$ 
  - Ex: probability calculation

# Calculating $P(s)$ : $s$ is a sentence

- Let  $s = w_1 \dots w_n$

$$\begin{aligned} P(w_1 \dots w_n) &= P(\text{BOS } w_1 \dots w_n \text{ EOS}) \\ &= P(w_1 | \text{BOS}) * P(w_2 | \text{BOS}, w_1) * \dots \\ &\quad P(w_n | w_{n-2}, w_{n-1}) * P(\text{EOS} | w_{n-1} w_n) \end{aligned}$$

If a n-gram contains a unknown word,  
skip the n-gram (i.e., remove it from the Eq)  
oov\_num ++;

=> number of ngrams when calculating  $P(s)$ :  $\text{sent\_leng} + 1 - \text{oov\_num}$

# Calculating Perplexity

$$PPL(T) = 10^{-\frac{1}{N} \lg P(T)}$$

Suppose T consists of m sentences:  $s_1, \dots, s_m$

$$\lg P(T) = \lg \prod_{i=1}^m P(s_i) = \sum_{i=1}^m \lg P(s_i)$$

$$N = \text{word\_num} + \text{sent\_num} - \text{oov\_num}$$

# Some intuition about perplexity

- Given a vocabulary  $V$  and assume uniform distribution; i.e.,  $P(w) = 1/|V|$
- The perplexity of any test data  $T$  with unigram LM is:

$$PPL(T) = P(T)^{-\frac{1}{N}} = \left(\frac{1}{|V|}\right)^N)^{-\frac{1}{N}} = \frac{1}{|V|}^{N * (-\frac{1}{N})} = |V|$$

- Perplexity is a measure of effective “branching factor”.

# Entropy and perplexity

- Entropy measures the information content in a distribution == the uncertainty
  - If I can predict the next word before it comes, there's no information content
  - Zero uncertainty means the signal has zero information
  - How many bits of additional information do I need to guess the next symbol?
- Perplexity is the average branching factor
  - If message has zero information, then branching factor is 1
  - If message needs one bit, branching factor is 2
  - If message needs two bits, branching factor is 4
- Entropy and perplexity measure the same thing (uncertainty / information content) with different scales

# Language model perplexity

- Recipe:
  - Train a language model on training data
  - Get negative logprobs of test data, compute average
  - Exponentiate!
- Perplexity correlates rather well with:
  - Speech recognition error rates
  - MT quality metrics
- LM Perplexities for word-based models are normally between say 50 and 1000
- Need to drop perplexity by a significant ***fraction*** (not absolute amount) to make a visible impact

# Standard metrics for LM

- Direct evaluation:
  - \_ Perplexity
- Indirect evaluation:
  - \_ ASR
  - \_ MT
  - \_ ...

# ASR

- Word error rate (WER):
  - \_ System: And he saw apart of the movie
  - \_ Gold: Andy saw a part of the movie
  - $WER = 3/7$
- Calculate WER with different LMs.



# Outline

- Motivation: LM applications
- N-gram LM
- Evaluation
- Other models and adaptation

# Incorporating Longer Distance Context

- Why use longer context?
  - N-grams are approximation
    - Model size
    - Sparseness
- What sorts of information in longer context?
  - Priming
  - Topic
  - Sentence type
  - Dialogue act
  - Syntax

# Long Distance LMs

- Bigger n!
  - 284M words:  $\leq 6$ -grams improve; 7-20 no better
- Topic models:
  - Intuition: Text is about some topic, on-topic words likely
    - $P(w \mid h) = \sum_t P(w, t \mid h)$   
 $\sim \sum_t P(w \mid t) P(t \mid h)$
- Class-based LM
- Neural LM: use embeddings for words and larger context
- ...

# Class-Based Language Model

- Variant of n-gram models using classes or clusters
- Motivation: Sparseness
  - \_ Flight app.:  $P(\text{ORD} \mid \text{to}), P(\text{JFK} \mid \text{to}), \dots P(\text{airport\_name} \mid \text{to})$ 
    - Relate probability of n-gram to word classes & class ngram
- IBM clustering: assume each word in single class
  - \_  $P(w_i \mid w_{i-1}) \sim P(c_{i-1} \mid w_{i-1}) P(c_i \mid c_{i-1}) P(w_i \mid c_i)$
  - \_ Learn by MLE from data
- Where do classes come from?
  - \_ Hand-designed for application (e.g. ATIS)
  - \_ Automatically induced clusters from corpus

# LM Adaptation

- Challenge: Need LM for a new domain
  - \_ Have little in-domain data
- Intuition: Much of language is pretty general
  - \_ Can build from a 'general' LM + in-domain data
- Approach: LM adaptation
  - \_ Train on a large domain-independent corpus
  - \_ Adapt that with a small in-domain data set
- What large corpus?
  - \_ Web counts! e.g., Google n-grams

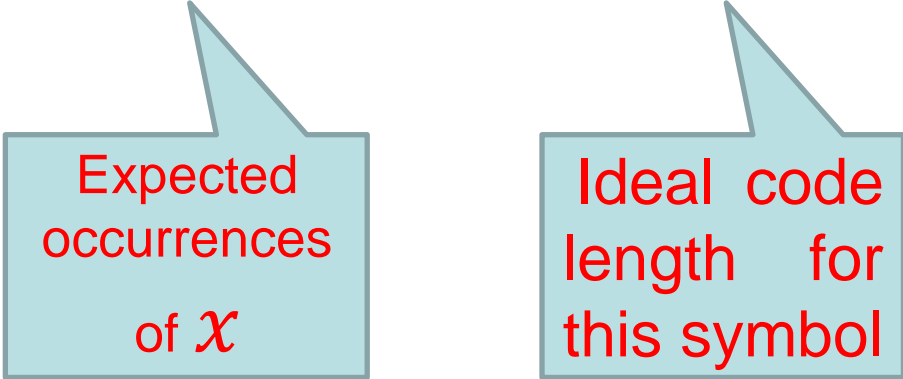
# Summary

- N-gram LM is the most common way to build a LM.
- Evaluation for LM:
  - Perplexity =  $10^{-1/N * \lg P(T)} = 2^{H(L,P)}$
  - \_ Indirect measures: WER for ASR, BLEU for MT, etc.
- Other LMs: class-based LM, structured LM, neural LM

# Additional slides

# Computing Entropy

- $H(x) = \sigma_x(p(x) \times (-\log_2 p(x)))$



Expected  
occurrences  
of  $x$

Ideal code  
length for  
this symbol



# Entropy

- Entropy is a measure of the uncertainty associated with a distribution.

$$H(X) = -\sum_x p(x) \log p(x)$$

- The lower bound on the number of bits it takes to transmit messages.
- An example:
  - Display the results of horse races.
  - Goal: minimize the number of bits to encode the results.

# An example

- Uniform distribution:  $p_i = 1/8$ .

$$H(X) = - \sum_{i=1}^8 p_i \log_2 p_i = 3 \text{ bits}$$

$$8 * \left( \frac{1}{8} \log_2 \frac{1}{8} \right)$$

- Non-uniform distribution:  $(1/2, 1/4, 1/8, 1/16, 1/64, 1/64, 1/64, 1/64)$

$$H(X) = -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{4} \log_2 \frac{1}{4} + \frac{1}{8} \log_2 \frac{1}{8} + \frac{1}{16} \log_2 \frac{1}{16} + 4 * \frac{1}{64} \log_2 \frac{1}{64}\right) = 2 \text{ bits}$$

$(0, 10, 110, 1110, 111100, 111101, 111110, 111111)$

➔ Uniform distribution has higher entropy.

➔ MaxEnt: make the distribution as “uniform” as possible.

# Cross Entropy

- Entropy: 
$$H(X) = -\sum_x p(x) \log p(x)$$
- Cross Entropy: 
$$H_c(X) = -\sum_x p(x) \log q(x)$$
- Cross entropy is a distance measure between  $p(x)$  and  $q(x)$ :  $p(x)$  is the true probability;  $q(x)$  is our estimate of  $p(x)$ .

$$H_c(X) \geq H(X)$$

# Cross entropy, formally

$$H(p) = \sum_x p(x) \times (-\log_2 p(x))$$

$$H(p, q) = \sum_x p(x) \times (-\log_2 q(x))$$

True distribution  $p(x)$  assumed distribution  $q(x)$

Wrote codebook using  $q(x)$  encode messages from  $p(x)$

Let  $p(x)$  be count-based distribution of test data  $w_1^n$ , then

$$\begin{aligned} \sum_{i=1}^n \frac{1}{n} \times (-\log_2 q(w_i)) &= \sum_x \frac{c(x \in w_1^n)}{n} \times (-\log_2 q(x)) \\ &= \sum_x p(x) \times (-\log_2 q(x)) \end{aligned}$$

# Cross entropy of a language

- The cross entropy of a language  $L$ :

$$H(L, q) = -\lim_{n \rightarrow \infty} \frac{\sum_{x_{1n}} p(x_{1n}) \log q(x_{1n})}{n}$$

- If we make certain assumptions that the language is “nice”, then the cross entropy can be calculated as:

$$H(L, q) = -\lim_{n \rightarrow \infty} \frac{\log q(x_{1n})}{n} \approx -\frac{\log q(x_{1n})}{n}$$

# Perplexity

$$PPL(T) = P(T)^{-\frac{1}{N}} = \frac{1}{\sqrt[N]{P(T)}}$$

$$= 2^{-\frac{1}{N} \log_2 P(T)}$$

$$= 2^{H(L, P)}$$