

# NLP: N-Grams

Dan Garrette  
dhg@cs.utexas.edu

September 21, 2013

## 1 Language Modeling Tasks

- Language identification / Authorship identification
- Machine Translation
- Speech recognition
- Optical character recognition (OCR)
- Context-sensitive spelling correction
- Predictive text (text messaging clients, search engines, etc)
- Generating spam
- Code-breaking (e.g. decipherment)

## 2 Kinds of Language Models

- Bag of words
- Sequences of words
- Sequences of tagged words
- Grammars
- Topic models

## 3 Language as a sequence of words

- What is the probability of seeing a particular sequence of words?
- Simplified model of language: words in order
- $p(\text{some words in sequence})$

- $p(\text{“an interesting story”})$  ?
- $p(\text{“a interesting story”})$  ?
- $p(\text{“a story interesting”})$  ?
- $p(\text{next} \mid \text{some sequence of previous words})$ 
  - $p(\text{“austin”} \mid \text{“university of texas at”})$  ?
  - $p(\text{“dallas”} \mid \text{“university of texas at”})$  ?
  - $p(\text{“giraffe”} \mid \text{“university of texas at”})$  ?

Use as a language model

- Language ID: This sequence is most likely to be seen in what language?
- Authorship ID: What is the most likely person to have produced this sequence of words?
- Machine Translation: Which sentence (word choices, ordering) seems most like the target language?
- Text Prediction: What’s the most likely next word in the sequence

Notation

- Probability of a sequence is a joint probability of words
- But the order of the words matters, so each gets its own feature
- $p(\text{“an interesting story”}) = p(w_{-3} = \text{an}, w_{-2} = \text{interesting}, w_{-1} = \text{story})$
- $p(\text{“austin”} \mid \text{“university of texas at”})$   
 $= p(w_0 = \text{austin} \mid w_{-4} = \text{university}, w_{-3} = \text{of}, w_{-2} = \text{texas}, w_{-1} = \text{at})$
- So  $w_0$  means “current word” and  $w_{-1}$  is “the previous word”.
- We will typically use the short-hand

## 4 Estimating Sequence Probabilities

- To build a statistical model, we need to set parameters.
- Our parameters: probabilities of various sequences of text
- Maximum Likelihood Estimate (MLE): of all the sequences of length N, what proportion are the relevant sequence?
- $p(\text{university of texas at austin})$   
 $= p(w_{-5} = \text{university}, w_{-4} = \text{of}, w_{-3} = \text{texas}, w_{-2} = \text{at}, w_{-1} = \text{austin})$   
 $= \frac{C(\text{“university of texas at austin”})}{C(\text{all 5-word sequences})} = \frac{3}{25,000,000}$

- $p(a \text{ bank}) = \frac{C(\text{"a bank"})}{C(\text{all 2-word sequences})} = \frac{609}{50,000,000}$
- $p(in \text{ the}) = \frac{C(\text{"in the"})}{C(\text{all 2-word sequences})} = \frac{312,776}{50,000,000}$
- Long sequences are unlikely to have any counts:  
 $p(\text{the university of texas football team started the season off right by scoring a touchdown in the final seconds of play to secure a stunning victory over the out-of-town challengers})$   
 $= \frac{C(\dots \text{that sentence } \dots)}{C(\text{all 30-word sequences})} = \mathbf{0.0}$
- Even shorter sentences may not have counts, even if they make sense, are perfectly grammatical, and not improbable that someone might say them:
  - “*university of texas in amarillo*”
- We need a way of estimating the probability of a long sequence, even though counts will be low.

Make a “naïve” assumption?

- With naïve Bayes, we dealt with this problem by assuming that all features were independent.
- $p(w_{-5} = \text{university}, w_{-4} = \text{of}, w_{-3} = \text{texas}, w_{-2} = \text{in}, w_{-1} = \text{amarillo}) = p(w = \text{university}) \cdot p(w = \text{of}) \cdot p(w = \text{texas}) \cdot p(w = \text{in}) \cdot p(w = \text{amarillo})$
- But this loses the difference between “university of texas in amarillo”, which seems likely, and “university texas amarillo in of”, which does not
- This amounts to a “bag of words” model

Use Chain Rule?

- Long sequences are sparse, short sequences are less so
- Break down long sequences using the chain rule
- $p(\text{university of texas in amarillo}) = p(\text{university}) \cdot p(\text{of} \mid \text{university}) \cdot p(\text{texas} \mid \text{university of}) \cdot p(\text{in} \mid \text{university of texas}) \cdot p(\text{amarillo} \mid \text{university of texas in})$
- “p(seeing ‘university’) times p(seeing ‘of’ given that the previous word was ‘university’) times p(seeing ‘texas’ given that the previous two words were ‘university of’) ...”
- $p(\text{university}) = \frac{C(\text{university})}{\sum_x C(x)} = \frac{C(\text{university})}{C(\text{all words})}$ ; easy to estimate  
 $p(\text{of} \mid \text{university}) = \frac{C(\text{university of})}{\sum_w C(\text{university } w)} = \frac{C(\text{university of})}{C(\text{university})}$ ; easy to estimate  
 $p(\text{texas} \mid \text{university of}) = \frac{C(\text{university of texas})}{\sum_w C(\text{university of } w)} = \frac{C(\text{university of texas})}{C(\text{university of})}$ ; easy to estimate  
 $p(\text{in} \mid \text{university of texas}) = \frac{C(\text{university of texas in})}{\sum_w C(\text{university of texas } w)} = \frac{C(\text{university of texas in})}{C(\text{university of texas})}$ ; easy to estimate  
 $p(\text{amarillo} \mid \text{university of texas in}) = \frac{C(\text{university of texas in amarillo})}{\sum_w C(\text{university of texas in } w)} = \frac{C(\text{university of texas in amarillo})}{C(\text{university of texas in})}$ ;  
**same problem**
- So this doesn’t help us at all.

## 5 N-Grams

- We don't necessarily want a "fully naïve" solution
  - Partial independence: limit how far back we look
- "Markov assumption": future behavior depends only on recent history
  - $k^{th}$ -order Markov model: depend only on  $k$  most recent states
- **n-gram**: sequence of  $n$  words
- **n-gram model**: statistical model of word sequences using n-grams.
- $p(\textit{university of texas in amarillo})$ 
  - 5+ -gram:  

$$p(\textit{university}) \cdot p(\textit{of} \mid \textit{university}) \cdot p(\textit{texas} \mid \textit{university of}) \cdot p(\textit{in} \mid \textit{university of texas}) \cdot p(\textit{amarillo} \mid \textit{university of texas in})$$
  - 3-gram (trigram):  

$$p(\textit{university}) \cdot p(\textit{of} \mid \textit{university}) \cdot p(\textit{texas} \mid \textit{university of}) \cdot p(\textit{in} \mid \textit{of texas}) \cdot p(\textit{amarillo} \mid \textit{texas in})$$
  - 2-gram (bigram):  

$$p(\textit{university}) \cdot p(\textit{of} \mid \textit{university}) \cdot p(\textit{texas} \mid \textit{of}) \cdot p(\textit{in} \mid \textit{texas}) \cdot p(\textit{amarillo} \mid \textit{in})$$
  - 1-gram (unigram) / bag-of-words model / full independence:  

$$p(\textit{university}) \cdot p(\textit{of}) \cdot p(\textit{texas}) \cdot p(\textit{in}) \cdot p(\textit{amarillo})$$
- Idea: reduce necessary probabilities to an estimatable size.
- Estimating bigrams:  

$$p(\textit{university}) \cdot p(\textit{of} \mid \textit{university}) \cdot p(\textit{texas} \mid \textit{of}) \cdot p(\textit{in} \mid \textit{texas}) \cdot p(\textit{amarillo} \mid \textit{in}) =$$

$$\frac{C(\textit{university})}{\sum_x C(x)} \cdot \frac{C(\textit{university of})}{\sum_x C(\textit{university } x)} \cdot \frac{C(\textit{of texas})}{\sum_x C(\textit{of } x)} \cdot \frac{C(\textit{texas in})}{\sum_x C(\textit{texas } x)} \cdot \frac{C(\textit{in amarillo})}{\sum_x C(\textit{in } x)} =$$

$$\frac{C(\textit{university})}{C(\textit{all words})} \cdot \frac{C(\textit{university of})}{C(\textit{university})} \cdot \frac{C(\textit{of texas})}{C(\textit{of})} \cdot \frac{C(\textit{texas in})}{C(\textit{texas})} \cdot \frac{C(\textit{in amarillo})}{C(\textit{in})}$$
- All of these should be easy to estimate!

## 6 N-Gram Model of Sentences

- Sentences are sequences of words, but with starts and ends.
- We also want to model the likelihood of words being at the beginning/end of a sentence.
- Append special "words" to the sentence
  - $n-1$   $\langle b \rangle$  symbols to beginning
  - only one  $\langle \backslash b \rangle$  to then end needed
    - \*  $p(\langle \backslash b \rangle \mid \cdot \langle \backslash b \rangle) = 1.0$  since  $\langle \backslash b \rangle$  would always be followed by  $\langle \backslash b \rangle$ .

- “the man walks the dog .” (trigrams)
  - Becomes “ $\langle b \rangle \langle b \rangle$  the man walks the dog .  $\langle \backslash b \rangle$ ”
  - $p(\langle b \rangle \langle b \rangle \text{ the man walks the dog . } \langle \backslash b \rangle) =$ 

$$\begin{aligned}
& p(\text{the} \mid \langle b \rangle \langle b \rangle) \\
& \cdot p(\text{man} \mid \langle b \rangle \text{ the}) \\
& \cdot p(\text{walks} \mid \text{the man}) \\
& \cdot p(\text{the} \mid \text{man walks}) \\
& \cdot p(\text{dog} \mid \text{walks the}) \\
& \cdot p(. \mid \text{the dog}) \\
& \cdot p(\langle \backslash b \rangle \mid \text{dog} .)
\end{aligned}$$
- Can be generalized to model longer texts: paragraphs, documents, etc:
  - Good: can model ngrams that cross sentences (e.g.  $p(w_0 \mid .)$  or  $p(w_0 \mid ?)$ )
  - Bad: more sparsity on  $\langle b \rangle$  and  $\langle \backslash b \rangle$

## 7 Sentence Likelihood Examples

Example dataset:

```

<b> the dog runs . </b>
<b> the dog walks . </b>
<b> the man walks . </b>
<b> a man walks the dog . </b>
<b> the cat walks . </b>
<b> the dog chases the cat . </b>

```

Sentence Likelihood with Bigrams:

$$\begin{aligned}
& p(\langle b \rangle \text{ the dog walks . } \langle \backslash b \rangle) \\
& = p(\text{the} \mid \langle b \rangle) \cdot p(\text{dog} \mid \text{the}) \cdot p(\text{walks} \mid \text{dog}) \cdot p(. \mid \text{walks}) \cdot p(\langle \backslash b \rangle \mid .) \\
& = \frac{C(\langle b \rangle \text{ the})}{\sum_x C(\langle b \rangle x)} \cdot \frac{C(\text{the dog})}{\sum_x C(\text{the } x)} \cdot \frac{C(\text{dog walks})}{\sum_x C(\text{dog } x)} \cdot \frac{C(\text{walks} .)}{\sum_x C(\text{walks } x)} \cdot \frac{C(. \langle \backslash b \rangle)}{\sum_x C(. x)} \\
& = \frac{C(\langle b \rangle \text{ the})}{C(\langle b \rangle)} \cdot \frac{C(\text{the dog})}{C(\text{the})} \cdot \frac{C(\text{dog walks})}{C(\text{dog})} \cdot \frac{C(\text{walks} .)}{C(\text{walks})} \cdot \frac{C(. \langle \backslash b \rangle)}{C(.)} \\
& = \frac{5}{6} \cdot \frac{4}{7} \cdot \frac{1}{4} \cdot \frac{3}{4} \cdot \frac{6}{6} = 0.83 \cdot 0.57 \cdot 0.25 \cdot 0.75 \cdot 1.0 = \mathbf{0.089}
\end{aligned}$$

$$\begin{aligned}
& p(\langle b \rangle \text{ the cat walks the dog . } \langle \backslash b \rangle) \\
& = p(\text{the} \mid \langle b \rangle) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{walks} \mid \text{cat}) \cdot p(\text{the} \mid \text{walks}) \cdot p(\text{dog} \mid \text{the}) \cdot p(. \mid \text{dog}) \cdot p(\langle \backslash b \rangle \mid .) \\
& = \frac{C(\langle b \rangle \text{ the})}{\sum_x C(\langle b \rangle x)} \cdot \frac{C(\text{the cat})}{\sum_x C(\text{the } x)} \cdot \frac{C(\text{cat walks})}{\sum_x C(\text{cat } x)} \cdot \frac{C(\text{walks the})}{\sum_x C(\text{walks } x)} \cdot \frac{C(\text{the dog})}{\sum_x C(\text{the } x)} \cdot \frac{C(\text{dog} .)}{\sum_x C(\text{dog } x)} \cdot \frac{C(. \langle \backslash b \rangle)}{\sum_x C(. x)} \\
& = \frac{C(\langle b \rangle \text{ the})}{C(\langle b \rangle)} \cdot \frac{C(\text{the cat})}{C(\text{the})} \cdot \frac{C(\text{cat walks})}{C(\text{cat})} \cdot \frac{C(\text{walks the})}{C(\text{walks})} \cdot \frac{C(\text{the dog})}{C(\text{the})} \cdot \frac{C(\text{dog} .)}{C(\text{dog})} \cdot \frac{C(. \langle \backslash b \rangle)}{C(.)} \\
& = \frac{5}{6} \cdot \frac{1}{7} \cdot \frac{1}{1} \cdot \frac{1}{4} \cdot \frac{4}{7} \cdot \frac{1}{4} \cdot \frac{6}{6} = 0.83 \cdot 0.14 \cdot 1.0 \cdot 0.25 \cdot 0.57 \cdot 0.25 \cdot 1.0 = \mathbf{0.004}
\end{aligned}$$

$$\begin{aligned}
& p(\langle b \rangle \text{ the cat runs } . \ \langle \backslash b \rangle) \\
&= p(\text{the} \mid \langle b \rangle) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{runs} \mid \text{cat}) \cdot p(. \mid \text{runs}) \cdot p(\langle \backslash b \rangle \mid .) \\
&= \frac{C(\langle b \rangle \text{ the})}{\sum_x C(\langle b \rangle x)} \cdot \frac{C(\text{the cat})}{\sum_x C(\text{the } x)} \cdot \frac{C(\text{cat runs})}{\sum_x C(\text{cat } x)} \cdot \frac{C(\text{runs } .)}{\sum_x C(\text{runs } x)} \cdot \frac{C(. \ \langle \backslash b \rangle)}{\sum_x C(. \ x)} \\
&= \frac{C(\langle b \rangle \text{ the})}{C(\langle b \rangle)} \cdot \frac{C(\text{the cat})}{C(\text{the})} \cdot \frac{C(\text{cat runs})}{C(\text{cat})} \cdot \frac{C(\text{runs } .)}{C(\text{runs})} \cdot \frac{C(. \ \langle \backslash b \rangle)}{C(.)} \\
&= \frac{5}{6} \cdot \frac{1}{7} \cdot \frac{0}{1} \cdot \frac{1}{1} \cdot \frac{6}{6} = 0.83 \cdot 0.14 \cdot \mathbf{0.0} \cdot 1.0 \cdot 1.0 = \mathbf{0.000}
\end{aligned}$$

- Longer sentences have lower likelihoods.
  - This makes sense because longer sequences are harder to match exactly.
- Zeros happen when an n-gram isn't seen.

## 8 Handling Sparsity

How big of a problem is sparsity?

- Alice's Adventures in Wonderland
  - Vocabulary (all word types) size:  $V = 3,569$
  - Distinct bigrams: 17,149;  $\frac{17,149}{|V|^2}$ , or 99.8% of possible bigrams unseen
  - Distinct trigrams: 28,540;  $\frac{17,149}{|V|^3}$ , or 99.9999994% of possible trigrams unseen
- If a sequence contains an unseen ngram, it will have likelihood zero: an impossible sequence.
- Many legitimate ngrams will simply be absent from the corpus.
- This does not mean they are impossible.
- Even ungrammatical/nonsense ngrams should not cause an entire sequence's likelihood to be zero.
- Many others will be too infrequent to estimate well.

Add- $\lambda$  Smoothing

- Add some constant  $\lambda$  to every count, including unseen ngrams
- $V$  is the Vocabulary — all word types — including  $\langle \backslash b \rangle$  (if necessary  $n > 1$ )
- $p(w_0 \mid w_{1-n} \dots w_{-1}) = \frac{C(w_{1-n} \dots w_{-1} w_0) + 1}{\sum_x (C(w_{1-n} \dots w_{-1} x) + 1)} = \frac{C(w_{1-n} \dots w_{-1} w_0) + 1}{(\sum_x C(w_{1-n} \dots w_{-1} x)) + |V|} = \frac{C(w_{1-n} \dots w_{-1} w_0) + 1}{C(w_{1-n} \dots w_{-1}) + |V|}$
- Add  $|V|$  to the denominator to account for the fact that there is an extra count for every  $x$
- In practice it over-smoothes, even when  $\lambda < 1$

Good-Turing Smoothing

- Estimate counts of things you haven't seen from counts of things you have
- Estimate probability of things which occur  $c$  times with the probability of things which occur  $c + 1$  times
- $c^* = (c + 1) \frac{N_{c+1}}{N_c}$
- $p_{GT}^*(things\ with\ freq\ 0) = \frac{N_1}{N}$
- $p_{GT}(w_0 \mid w_{1-n} \dots w_{-1}) = \frac{C^*(w_{1-n} \dots w_{-1} w_0)}{C^*(w_{1-n} \dots w_{-1})}$

### Stupid Backoff

- If  $p(n\text{-gram})=0$ , use  $p((n-1)\text{-gram})$
- Works shockingly well for huge datasets

### Interpolation

- Mix  $n$ -gram probability with probabilities from lower-order models

$$\begin{aligned}\hat{p}(w_0 \mid w_{-2} w_{-1}) &= \lambda_3 \cdot p(w_0 \mid w_{-2} w_{-1}) \\ &= \lambda_2 \cdot p(w_0 \mid w_{-1}) \\ &= \lambda_1 \cdot p(w_0)\end{aligned}$$

- $\lambda_i$  terms used to decide how much to smooth
- $\sum_i \lambda_i = 1$ , because they are proportions
- Use *dev* dataset to tune  $\lambda$  hyperparameters
- Also useful for combining models trained on different data:
  - Can interpolate “customized” models with “general” models
  - Baseline English + regional English + user-specific English
  - Little in-domain data, lots of out-of-domain

### Kneser-Ney Smoothing

- Intuition: interpolate based on “openness” of the context
- Words seen in more contexts are more likely to appear in others
- Even if we haven't seen  $w_0$  following the context, if the context is “open” (supports a wide variety of “next words”), then it is more likely to support  $w_0$
- Boost counts based on  $|\{x : C(w_{1-n} \dots w_{-1} x) > 0\}|$ , the number of different “next words” seen after  $w_{1-n} \dots w_{-1}$

## 9 Out-of-Vocabulary Words (OOV)

Add- $\lambda$

- If ngram contains OOV item, assume count of  $\lambda$ , just like for all other ngrams.
- Probability distributions become invalid. We can't know the full vocabulary size, so we can't normalize counts correctly.

$\langle unk \rangle$

- Create special token  $\langle unk \rangle$
- Create a fixed lexicon  $L$ 
  - All types in some subset of training data?
  - All types appearing more than  $k$  times?
- $V = L + \langle unk \rangle$ ,  $|V| = |L| + 1$
- Before training, change any word not in  $L$  to  $\langle unk \rangle$
- Then train as usual as if  $\langle unk \rangle$  was a normal word
- For new sentence, again replace words not in  $L$  with  $\langle unk \rangle$  before using model
- Probabilities containing  $\langle unk \rangle$  measure likelihood with *some rare word*
- Problem: the “rare” word is no longer rare since there are many  $\langle unk \rangle$  tokens
  - Ngrams with  $\langle unk \rangle$  will have higher probabilities than those with any particular rare word
  - Not so bad when comparing same sequence under multiple models. All will have inflated probabilities.
  - More problematic when comparing probabilities of different sequences under the same model
    - \*  $p(i \text{ **totes** know}) < p(i \text{ **totally** know}) < p(i \text{ **unk** know})$

## 10 Evaluation

Extrinsic

- Use the model in some larger task. See if it helps.
- More realistic
- Harder

Intrinsic



- Evaluate on a test corpus
- Easier

Perplexity

- Intrinsic measure of model quality
- How well does the model “fit” the test data?
- How “perplexed” is the model whe it sees the test data?
- Measure the probability of the test corpus, normalize for number of words.
- $PP(W) = \frac{1}{|W|} \sqrt[|W|]{\frac{1}{p(w_1 w_2 \dots w_{|W|})}}$

## 11 Limitations

- Long-range dependencies

## 12 Generative Models

- Generative models are designed to model how the data *could have been generated*.
- The best parameters are those that would most likely generate the data.
- MLE maximizes that likelihood that the training data was generated by the model.
- As such, we can *actually generate* data from a model.
- Trigram model:
  - General:
 

For each sequence:

    1. Sample a word  $w_0$  according to  $w_0 \sim p(w_0)$
    2. Sample a second word  $w_1$  according to  $w_1 \sim p(w_1 \mid w_0)$
    3. Sample a next word  $w_k$  according to  $w_k \sim p(w_k \mid w_{k-2} w_{k-1})$
    4. Repeat step 3 until you feel like stopping.
  - Sentences:
 

For each sentence:

    1. Sample a word  $w_0$  according to  $w_0 \sim p(w_0 \mid \langle b \rangle \langle b \rangle)$
    2. Sample a second word  $w_1$  according to  $w_1 \sim p(w_1 \mid w_0 \langle b \rangle)$
    3. Sample a next word  $w_k$  according to  $w_k \sim p(w_k \mid w_{k-2} w_{k-1})$
    4. Repeat until  $\langle \backslash b \rangle$  is drawn.
- Longer  $n$  generates more coherent text

- Too-large  $n$  just ends up generating sentences from the training data because most counts will be 1 (no choice of next word).
- Naïve Bayes was a generative model too!

For each instance:

1. Sample a label  $l$  according to  $l \sim p(\text{Label} = l)$
  2. For each feature  $F$ : sample a value  $v$  according to  $v \sim p(F = v \mid \text{Label} = l)$
- We will see many more generative models throughout this course

## 13 Citations

Some content adapted from:

- [http://courses.washington.edu/ling570/gina\\_fall11/slides/ling570\\_class8\\_smoothing.pdf](http://courses.washington.edu/ling570/gina_fall11/slides/ling570_class8_smoothing.pdf)