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
- $p$(some words in sequence)
- \( p(\text{"an interesting story"}) \) ?
- \( p(\text{"a interesting story"}) \) ?
- \( p(\text{"a story interesting"}) \) ?

- \( p(\text{next | some sequence of previous words}) \)
  - \( p(\text{"austin" | "university of texas at"}) \) ?
  - \( p(\text{"dallas" | "university of texas at"}) \) ?
  - \( p(\text{"giraffe" | "university of texas at"}) \) ?

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
- \( p(\text{"an interesting story"}) = p(w_1 = an, \ w_2 = interesting, \ w_3 = story) \)
  - We will abbreviate this as \( p(\text{an interesting story}) \), but remember that the order matters, and that the words should be thought of as separate features
- \( p(\text{"austin" | "university of texas at"}) \)
  - \( = p(w_0 = austin \ | \ w_{-4} = university, \ w_{-3} = of, \ w_{-2} = texas, \ w_{-1} = at) \)
    - So \( w_0 \) means “current word” and \( w_{-1} \) is “the previous word”.
    - We will abbreviate this as \( p(\text{austin | university of texas at}) \), but remember that the order matters, and that the words should be thought of as separate features
- We will typically use the short-hand

4 Counting Words

- Terminology:
  - **Type**: Distinct word
  - **Token**: Particular occurrence of a word
  - “the man saw the saw”: 3 types, 5 tokens
What is a word? What do we count?

- Punctuation? Separate from neighboring words? Keep it at all?
- Stopwords?
- Lowercase everything?
- Distinct numbers vs. \( \langle \text{number} \rangle \)
- Hyphenated words?
- Lemmas only?
- Disfluencies? (um, uh)

Day 2

5 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?
- \( p(\text{university of texas at austin}) \)
  \[ p(w_1 = \text{university}, w_2 = \text{of}, w_3 = \text{texas}, w_4 = \text{at}, w_5 = \text{austin}) = \frac{C(\text{university of texas at austin})}{C(\text{all 5-word sequences})} = \frac{3}{25,000,000} \]

- \( p(\text{a bank}) = \frac{C(\text{a bank})}{C(\text{all 2-word sequences})} = \frac{609}{50,000,000} \)

- \( p(\text{in the}) = \frac{C(\text{in the})}{C(\text{all 2-word sequences})} = \frac{312,776}{50,000,000} \)

- Long sequences are unlikely to have any counts:
  \[ p(\text{the university of texas football team started the season off right by scoring a touchdown in the final seconds of play to secure a stunning victory over the out-of-town challengers}) = \frac{C(\text{... that sentence ...})}{C(\text{all 30-word sequences})} = 0.0 \]

- Even shorter sentences may not have counts, even if they make sense, are perfecty 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.
- \( p(w_1 = \text{university}, w_2 = \text{of}, w_3 = \text{texas}, w_4 = \text{in}, w_5 = \text{amarillo}) = \)
  \( p(w = \text{university}) \cdot p(w = \text{of}) \cdot p(w = \text{texas}) \cdot p(w = \text{in}) \cdot p(w = \text{amarillo}) \)
- 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’) ...
- \( p(\text{university}) = \frac{C(\text{university})}{\sum_{x \in V} C(x)} = \frac{C(\text{university})}{C(\text{all words})}; \text{easy to estimate} \)
- \( p(\text{of} \mid \text{university}) = \frac{C(\text{university of})}{\sum_{w} C(\text{university} \times w)} = \frac{C(\text{university of})}{C(\text{university})}; \text{easy to estimate} \)
- \( p(\text{texas} \mid \text{university of}) = \frac{C(\text{university of texas})}{\sum_{w} C(\text{university of} \times w)} = \frac{C(\text{university of texas})}{C(\text{university of})}; \text{easy to estimate} \)
- \( p(\text{in} \mid \text{university of texas}) = \frac{C(\text{university of texas in})}{\sum_{w} C(\text{university of texas in} \times w)} = \frac{C(\text{university of texas in})}{C(\text{university of texas})}; \text{easy to estimate} \)
- \( p(\text{amarillo} \mid \text{university of texas in}) = \frac{C(\text{university of texas in amarillo})}{\sum_{w} C(\text{university of texas in} \times w)} = \frac{C(\text{university of texas in amarillo})}{C(\text{university of texas in})}; \text{same problem} \)
- So this doesn’t help us at all.

6 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
  - \( k^{\text{th}}\)-order Markov model: depend only on \( k \) most recent states
- \textbf{n-gram}: sequence of \( n \) words
- \textbf{n-gram model}: statistical model of word sequences using n-grams.
Approximate all conditional probabilities by only looking back $n-1$ words (conditioning only on the previous $n-1$ words)

- For $n$, estimate everything in terms of: $p(w_n | w_1, w_2, ..., w_{n-1})$
- $p(w_T | w_1, w_2, ..., w_{T-1}) \approx p(w_T | w_{T-(n+1)}, ..., w_{T-1})$
- $p(\text{university of texas in amarillo})$
  - 5+ -gram:
    $p(\text{university}) \cdot p(\text{of} | \text{university}) \cdot p(\text{texas} | \text{university of}) \cdot p(\text{in} | \text{university of texas in})$
    \quad \cdot p(\text{amarillo} | \text{university of texas in})$
  - 3-gram (trigram):
    $p(\text{university}) \cdot p(\text{f} | \text{university}) \cdot p(\text{texas} | \text{university of}) \cdot p(\text{in} | \text{of})$
    \quad \cdot p(\text{amarillo} | \text{texas in})$
  - 2-gram (bigram):
    $p(\text{university}) \cdot p(\text{of} | \text{university}) \cdot p(\text{texas} | \text{of}) \cdot p(\text{in} | \text{texas}) \cdot p(\text{amarillo} | \text{in})$
  - 1-gram (unigram) / bag-of-words model / full independence:
    $p(\text{university}) \cdot p(\text{of}) \cdot p(\text{texas}) \cdot p(\text{in}) \cdot p(\text{amarillo})$

- Idea: reduce necessary probabilities to an estimatable size.

- Estimating trigrams:
  $p(\text{university}) \cdot p(\text{of} | \text{university}) \cdot p(\text{texas} | \text{university of}) \cdot p(\text{in} | \text{of})$
  \quad \cdot p(\text{amarillo} | \text{texas in})
  \quad = C(\text{university}) \sum_{x \in V} C(x) \cdot C(\text{university of}) \sum_{x \in V} C(\text{university} x) \cdot C(\text{university of texas}) \sum_{x \in V} C(\text{university of} x) \cdot C(\text{of} \sum_{x \in V} C(\text{of} x) \cdot C(\text{texas in amarillo}) \sum_{x \in V} C(\text{texas in} x)$

- All of these should be easy to estimate!

- Other advantages:
  - Smaller $n$ means fewer parameters to store. Means less memory required. Makes a difference on huge datasets or on limited memory devices (like mobile phones).

7 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
- $n-1$ ‘⟨S⟩’ symbols to beginning
- only one ‘⟨E⟩’ to then end needed
  $p(⟨E⟩ \mid ., ⟨E⟩) = 1.0$ since ⟨E⟩ would always be followed by ⟨E⟩.

“the man walks the dog .” (trigrams)
- Becomes “⟨S⟩ ⟨S⟩ the man walks the dog . ⟨E⟩”
- $p(⟨S⟩ ⟨S⟩ the man walks the dog . ⟨E⟩) = p(⟨S⟩ ⟨S⟩) \cdot p(\text{the} \mid ⟨S⟩ ⟨S⟩) \cdot p(\text{man} \mid ⟨S⟩ \text{the}) \cdot p(\text{walks} \mid \text{the man}) \cdot p(\text{the} \mid \text{man walks}) \cdot p(\text{dog} \mid \text{walks the}) \cdot p(\cdot \mid \text{the dog}) \cdot p(⟨E⟩ \mid \text{dog} .)$

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 ⟨S⟩ and ⟨E⟩

8 Sentence Likelihood Examples

Example dataset:

⟨S⟩ the dog runs . ⟨E⟩
⟨S⟩ the dog walks . ⟨E⟩
⟨S⟩ the man walks . ⟨E⟩
⟨S⟩ a man walks the dog . ⟨E⟩
⟨S⟩ the cat walks . ⟨E⟩
⟨S⟩ the dog chases the cat . ⟨E⟩

Sentence Likelihood with Bigrams:

$p(⟨S⟩ \text{ the dog walks . } ⟨E⟩)$

$= p(\text{the} \mid ⟨S⟩) \cdot p(\text{dog} \mid \text{the}) \cdot p(\text{walks} \mid \text{dog}) \cdot p(\cdot \mid \text{walks}) \cdot p(⟨E⟩ \mid .)$

$= \frac{C(⟨S⟩ \text{ the})}{\sum_{x \in V_{-⟨E⟩}} C(⟨S⟩ x)} \cdot \frac{C(\text{the dog})}{\sum_{x \in V} C(\text{the} x)} \cdot \frac{C(\text{dog walks})}{\sum_{x \in V} C(\text{dog} x)} \cdot \frac{C(⟨E⟩ \cdot)}{\sum_{x \in V} C(⟨E⟩ x)}$
\[
\sum_{x \in V} C(x) \cdot \sum_{x \in V} C(x) \cdot \sum_{x \in V} C(x) \cdot \sum_{x \in V} C(x) \cdot \sum_{x \in V} C(x) = 0.83 \cdot 0.57 \cdot 0.25 \cdot 0.75 \cdot 1.0 = 0.089
\]

\[
p(\langle S \rangle \text{ the cat walks the dog } . \langle E \rangle)
\]
\[
= p(\text{the } | \langle S \rangle) \cdot p(\text{cat } | \text{ the}) \cdot p(\text{walks } | \text{ cat}) \cdot p(\text{the } | \text{ walks}) \cdot p(\text{dog } | \text{ the}) \cdot p(\text{. } | \text{ dog}) \cdot p(\langle E \rangle | .)
\]
\[
= \frac{C(\langle S \rangle \text{ the})}{\sum_{x \in V} C(\langle S \rangle x)} \cdot \frac{C(\text{the cat})}{\sum_{x \in V} C(x)} \cdot \frac{C(\text{walks the})}{\sum_{x \in V} C(x)} \cdot \frac{C(\text{walks the})}{\sum_{x \in V} C(x)} \cdot \frac{C(\text{the dog})}{\sum_{x \in V} C(x)} \cdot \frac{C(\text{. dog .})}{\sum_{x \in V} C(x)}
\]
\[
= \frac{5}{6} \cdot \frac{1}{7} \cdot \frac{1}{1} \cdot \frac{4}{7} \cdot \frac{1}{4} \cdot \frac{6}{6} = 0.83 \cdot 0.14 \cdot 1.0 \cdot 0.25 \cdot 0.57 \cdot 0.25 \cdot 1.0 = 0.004
\]

\[
p(\langle S \rangle \text{ the cat runs } . \langle E \rangle)
\]
\[
= p(\text{the } | \langle S \rangle) \cdot p(\text{cat } | \text{ the}) \cdot p(\text{runs } | \text{ cat}) \cdot p(\text{. } | \text{ runs}) \cdot p(\langle E \rangle | .)
\]
\[
= \frac{C(\langle S \rangle \text{ the})}{\sum_{x \in V} C(\langle S \rangle x)} \cdot \frac{C(\text{the cat})}{\sum_{x \in V} C(x)} \cdot \frac{C(\text{runs the})}{\sum_{x \in V} C(x)} \cdot \frac{C(\text{runs .})}{\sum_{x \in V} C(x)} \cdot \frac{C(\langle E \rangle)}{\sum_{x \in V} C(x)}
\]
\[
= \frac{5}{6} \cdot \frac{1}{7} \cdot \frac{0}{1} \cdot \frac{1}{4} \cdot \frac{6}{6} = 0.83 \cdot 0.14 \cdot 0.0 \cdot 1.0 \cdot 1.0 = 0.000
\]

- 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.

\[\text{Day 3}\]

9 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; \( \frac{17,149}{\lfloor V \rfloor^2} \), or 99.8% of possible bigrams unseen
  - Distinct trigrams: 28,540; \( \frac{28,540}{\lfloor V \rfloor^3} \), or 99.9999994% of possible trigrams unseen
- If a sequence contains an unseen n-gram, 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-λ Smoothing

- Add some constant λ to every count, including unseen ngrams
- V is the Vocabulary — all possible “next word” types — including <E> (if necessary n > 1)
  - Don’t need <S> because it will never be the ‘next word’
- \( p(w_0 \mid w_{1-n} \ldots w_{-1}) = \frac{C(w_{1-n} \ldots w_{-1} w_0) + \lambda}{\sum_{x \in V} C(w_{1-n} \ldots w_{-1} x) + \lambda} \frac{C(w_{1-n} \ldots w_{-1} w_0) + \lambda}{(\sum_{x \in V} C(w_{1-n} \ldots w_{-1} x)) + \lambda |V|} = \frac{C(w_{1-n} \ldots w_{-1} w_0)}{C(w_{1-n} \ldots w_{-1} + \lambda |V|)} \)
  - Add |V| to denominator to account for the fact that there is an extra count for every x
- In practice it over-smoothes, even when \( \lambda < 1 \)
- Example dataset:
  - \(<S><S> the dog runs . <E><S><S> the dog walks . <E><S><S> the man walks . <E><S><S> a man walks the dog . <E><S><S> the cat walks . <E><S><S> the dog chases the cat . <E>\)
  - \( V = \{a, cat, chases, dog, man, runs, the, walks, .., <E>\} \)
  - \(|V| = 10\)
  - Sentence Likelihood with Trigrams:
    - \( p(\text{“the cat runs .”}) \)
    - \( = p(<S> <S> \text{the cat runs . } <E>) \)
    - \( = p(\text{the } | <S> <S> \text{ the}) \cdot p(\text{cat } | <S> <S> \text{ the}) \cdot p(\text{runs } | <S> <S> \text{ the}) \cdot p(\text{. } | <S> <S> \text{ the runs}) \cdot p(<E> | <S> <S> \text{ the runs}) \)
    - \( = \frac{C(<S> <S> \text{ the}) + 1}{\sum_{x \in V} C(<S> <S> x) + 1} \cdot \frac{C(<S> <S> \text{ the}) + 1}{\sum_{x \in V} C(<S> <S> x) + 1} \cdot \frac{C(\text{the cat runs}) + 1}{\sum_{x \in V} C(\text{the cat runs}) + 1} \cdot \frac{C(\text{cat runs .}) + 1}{\sum_{x \in V} C(\text{cat runs .}) + 1} \cdot \frac{C(\text{runs . } <E>)}{\sum_{x \in V} C(\text{runs . } x) + 1} \)
    - \( = \frac{5+1}{5+1} \cdot \frac{1+1}{1+1} \cdot \frac{0+1}{0+1} \cdot \frac{1+1}{1+1} \cdot \frac{0.13}{0.13} \cdot 0.08 \cdot 0.10 \cdot 0.18 = 0.000081 \)
- If the context was never seen, then the distribution is uniform:
  - \( p_{+\lambda}(w_0 \mid w_{-2} w_{-1}) = \frac{C(w_{-2} w_{-1} w_0) + \lambda}{(\sum_{x \in V} C(w_{-2} w_{-1} x)) + \lambda \cdot |V|} = \frac{0 + \lambda}{(\sum_{x \in V} 0) + \lambda \cdot |V|} = \frac{0 + \lambda \cdot |V|}{\lambda \cdot |V|} = \frac{1}{|V|} \)
• Since \(<S>\) can never be followed by \(<E>\)
  
  - \( p(<E> | <S> <S>) = 0 \)
  
  - The denominator \( \sum x \in V C(<S> <S> x) \) only gets |V|-1 smoothing counts

• \(<S>\) not included in V because we can’t transition to it.

• More smoothing on less common ngrams

  - With smoothing, we have counts for any possible type following “runs .”:
    
    \[
    \begin{align*}
    p(<E> | \text{runs .}) &= \frac{1+1}{1+10} = \frac{2}{11} = 0.18 \\
    p(\text{the} | \text{runs .}) &= \frac{0+1}{1+10} = \frac{1}{11} = 0.09
    \end{align*}
    \]
    
    * Counts of “runs .” are very low (only 1 occurrence), so estimates are bad
    * Bad estimates means more smoothing is good
    * MLE of “runs . <E>” is 1.0, add-1 smoothed becomes 0.18
      MLE of “runs . the” is 0.0, add-1 smoothed becomes 0.09
    * Original difference of 1.0 becomes difference of 0.09!
    * This makes sense because our estimates are so bad that we really can’t make a judgement about what could possibly follow “runs .”

  - Contexts with higher counts have less smoothing:
    
    \[
    \begin{align*}
    p(<E> | \text{walks .}) &= \frac{3+1}{3+10} = \frac{4}{13} = 0.31 \\
    p(\text{the} | \text{walks .}) &= \frac{0+1}{3+10} = \frac{1}{13} = 0.08
    \end{align*}
    \]
    
    * Counts of “walks .” are higher (3 occurrence), so estimates are better
    * Better estimates means less smoothing is good
    * MLE of “walks . <E>” is 1.0, add-1 smoothed becomes 0.31
      MLE of “walks . the” is 0.0, add-1 smoothed becomes 0.08
    * Original difference of 1.0 becomes difference of 0.23
    * Remains a larger gap than we saw for “runs . x”
    * This makes sense because our estimates are better, so we can be more sure that “walks .” should only be followed by \(<E>\)

• Disadvantages of add-\(\lambda\) smoothing:

  - Over-smoothes.

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

  \[
  c^* = (c + 1) \frac{N_{c+1}}{N_c}
  \]

  \[
  p_{GT}(\text{things with freq } 0) = \frac{N_1}{N}
  \]

  \[
  p_{GT}(w_0 | w_{1-n} \ldots w_{-1}) = \frac{C^*(w_{1-n} \ldots w_{-1} w_0)}{C^*(w_{1-n} \ldots w_{-1})}
  \]

9
Knesser-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 $|\{x : C(w_{1-n} \ldots w_{-1}x) > 0\}|$, the number of different “next words” seen after $w_{1-n} \ldots w_{-1}$

\[ \text{Day 4} \]

Interpolation

- Mix n-gram probability with probabilities from lower-order models
  \[
  \hat{p}(w_0 | w_{-2} w_{-1}) = \lambda_3 \cdot p(w_0 | w_{-2} w_{-1}) \\
  + \lambda_2 \cdot p(w_0 | w_{-1}) \\
  + \lambda_1 \cdot p(w_0)
  \]
- $\lambda_i$ terms used to decide how much to smooth
- $\sum_i \lambda_i = 1$ (still a valid probability distribution, 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

Stupid Backoff

- If $p(\text{n-gram})=0$, just use $p((\text{n-1})\text{-gram})$
- Does not yield a valid probability distribution
- Works shockingly well for huge datasets
10 Out-of-Vocabulary Words (OOV)

Add-λ

• If ngram contains OOV item, assume count of λ, 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.

⟨unk⟩

• Create special token ⟨unk⟩
• 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 ⟨unk⟩
• Then train as usual as if ⟨unk⟩was a normal word
• For new sentence, again replace words not in L with ⟨unk⟩before using model
• Probabilities containing ⟨unk⟩measure likelihood with some rare word
• Problem: the “rare” word is no longer rare since there are many ⟨unk⟩tokens
  – Ngrams with ⟨unk⟩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
    * \(p(\text{totes know}) < p(\text{totally know}) < p(\langle unk\rangle \text{ know})\)

11 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 when it sees the test data?
• Measure the probability of the test corpus, normalize for number of words.
• $PP(W) = \sqrt[|W|]{\prod_{w \in W} \frac{1}{p(w_1 w_2 \ldots w_{|W|})}}$

With individual sentences: $PP(s_1, s_2, \ldots) = \left( \sum_{i=1}^{|s_i|} \frac{1}{\prod_{i} p(s_i)} \right)$

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 $w_0 \sim p(w_0 \mid \langle S \rangle \langle S \rangle)$
    2. Sample a second word $w_1$ according to $w_1 \sim p(w_1 \mid w_0 \langle S \rangle)$
    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 $\langle E \rangle$ 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

$$
\begin{align*}
p(S \rightarrow \text{the}) &= 0.83 & p(\text{dog} \rightarrow \text{chases}) &= 0.25 \\
p(S \rightarrow a) &= 0.17 & p(\text{dog} \rightarrow \text{runs}) &= 0.25 \\
p(\text{the} \rightarrow \text{cat}) &= 0.29 & p(\text{dog} \rightarrow \text{walks}) &= 0.25 \\
p(\text{the} \rightarrow \text{dog}) &= 0.57 & p(\text{chases} \rightarrow \text{the}) &= 1.00 \\
p(\text{the} \rightarrow \text{man}) &= 0.14 & \\
p(a \rightarrow \text{man}) &= 1.00 & p(\text{runs} \rightarrow .) &= 1.00 \\
p(\text{man} \rightarrow \text{walks}) &= 1.00 & p(\text{walks} \rightarrow \text{the}) &= 0.25 \\
p(\text{cat} \rightarrow \text{walks}) &= 0.50 & p(\text{walks} \rightarrow .) &= 0.75 \\
p(\text{cat} \rightarrow .) &= 0.50 & p(\text{.} \rightarrow <E>) &= 1.00
\end{align*}
$$

13 How much data?

Choosing $n$
• Large $n$
  
  – More context for probabilities:
    $p$(phone) vs
    $p$(phone $|$ cell) vs
    $p$(phone $|$ your cell) vs
    $p$(phone $|$ off your cell) vs
    $p$(phone $|$ turn off your cell)

  – Long-range dependencies

• Small $n$
  
  – Better generalization
  – Better estimates
  – Long-range dependencies

How much training data?

• *As much as possible.*

• More data means better estimates

• Google N-Gram corpus uses 10-grams

(a) Data size matters more than algorithm (Banko and Brill, 2001)
(b) Results keep improving with more data (Norvig: Unreason-able ...)
(c) With enough data, stupid backoff approaches Knesser-Ney accuracy (Brants et al., 2007)

14 Citations

Some content adapted from: