Lecture 3: N-gram LMs revisited / Sequence Tagging: HMMs

Cornell Bowers CIS
Computer Science

Claire Cardie, Tanya Goyal

CS 4740 (and crosslists): Introduction to Natural Language Processing

Announcements

- HW0 due on Friday on Gradescope, 11.59 p.m.
- HW1 will be released Monday, Feb 3.
  - We will post a mega-thread on Ed to find a partner.
- Readings (Jurafsky & Martin, 3rd ed) are posted on the schedule (and are VERY VERY useful)
- Waitlist
  - non-CIS graduate students are unlikely to get into the class

Today

- Recap on n-gram language models (LMs)
- Learning n-gram models: an example
- Complications when building n-gram LMs
- Part-of-speech tagging
- HMMs for sequence tagging: introduction

What is a Language Model?

- A model that computes the probability of a word sequence:

$$P(w_1 w_2 w_3 ... w_n)$$

Mayenne

$$P(\textit{Mayenne ate my shoes today.}) = 10^{-12}$$

- A model that computes a probability distribution over possible next words:

$$P(w_n | w_1 w_2 w_3 ... w_{n-1})$$

$$P(\textit{today} \mid \textit{Mayenne ate my shoes}) = 10^{-3}$$

Language Modeling Problem

- Let $\mathcal{V}$ be a finite vocabulary of words.

$$\mathcal{V} = \{\text{ the, a, man, telescope, Madrid, two, } \ldots\}$$

- We can construct (infinite) word sequences $\mathbf{w}$

$$\mathcal{V}^{\dagger} = \{\text{ the, a, the a, the fan, the man, the man with a telescope}\}$$

- **Goal**: estimate a probability distribution $P(\mathbf{w})$ over **all** word sequences $\mathbf{w} \in \mathcal{V}^{\dagger}$
  - **Given**: a dataset of ***M*** word sequences (sentences) $\mathcal{D} = \{\mathbf{w}\}_{i=1}^{M}$

Terminology: the ambiguous term "word"

- We will often need to distinguish (the counting of)
  - **word types**
    - distinct words. The finite set, which you predetermine, is the *vocabulary* or *lexicon.*
  - **word tokens**
    - the words in the "running" text (instances of the vocab items).

Example: All for one and one for all .

  - **8 tokens** (if we choose to have punctuation in our lexicon)
  - **6 word types *if*** we assume capitalization is a distinguisher
  - **5 word types *if*** capitalization differences are ignored

Language Modeling Problem

- **Goal**: estimate a probability distribution $P(\mathbf{w})$ over **all** word sequences $\mathbf{w} \in \mathcal{V}^\dagger$
  - **Given**: a dataset of ***M*** word sequences (sentences) $\mathcal{D} = \{\mathbf{w}\}_{i=1}^{M}$

Näive option: compute the empirical distribution over the training data.

$$P(\mathbf{w}) =$$

Problems?

Does not generalize to unseen sequences, i.e. to valid ***w*** that do not appear in ***M***. Not enough data to gather reliable probabilities.

Language Modeling Problem

First, let's decompose $P(\mathbf{w})$

$$P(\mathbf{w}) = P(w_1 w_2 w_3 ... w_n)$$

applying chain rule

$$= P(w_1)P(w_2|w_1)P(w_3|w_2 w_1)...P(w_n|w_1 ... w_{n-1})$$

assumption: probability of a word depends on previous words only

$$= \prod_{i=1}^{n} P(w_i|w_1 ... w_{i-1})$$

$$P(I\ saw\ a\ man)$$
$$= P(I)P(saw\ /\ I)P(a\ /\ I\ saw)P(man\ /\ I\ saw\ a)$$

Language Modeling Problem

$$P(\mathbf{w}) = P(w_1 w_2 w_3 ... w_n) = \prod_{i=1}^{n} P(w_i | w_1 ... w_{i-1})$$

Can we now use count-based estimates?

$$= P(w_1) P(w_2|w_1) P(w_3|w_2 w_1) ... P(w_n|w_1 ... w_{n-1})$$

If a test sentence **w** is unseen in the training data, this will again be zero!

Language Modeling Problem

$$P(\mathbf{w}) = P(w_1 w_2 w_3 \ldots w_n) = \prod_{i=1}^{n} P(w_i | w_1 \ldots w_{i-1})$$

Key idea: Markov Assumption

Probability of each "next word" in a sequence only depends on a **fixed number** of previous words

Unigram model $\rightarrow$ $P(w_i | w_1 \ldots w_{i-1}) := P(w_i)$

Bigram model $\rightarrow$ $P(w_i | w_1 \ldots w_{i-1}) := P(w_i \mid w_{i-1})$

Trigram model $\rightarrow$ $P(w_i | w_1 \ldots w_{i-1}) := P(w_i \mid w_{i-2} w_{i-1})$

N-gram language models: Probability of each word depends on N-1 previous words.

Example

P(lost|Not all those who wander are)

According to our various models, that probability is equal to …

- Unigram model → P(lost)
- Bigram model → P(lost | are)
- Trigram model → P(lost | wander are)

Sequence probability according to a bigram LM

- Goal: compute $P(w_1\ w_2\ \ldots\ w_{n-1}\ w_n)$, with implicit w_0 := <s>

shorthand for

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$

$$\cong P(w_1)\ P(w_2|w_1)\ P(w_3|w_2)\ \ldots\ P(w_n|w_{n-1})$$

$$= P(w_1|\langle s\rangle)\ P(w_2|w_1)\ P(w_3|w_2)\ \ldots\ P(w_n|w_{n-1})$$

$$= \prod_{k=1}^{n} P(w_k|w_{k-1})$$

One way to "learn" an n-gram model

- "Raw count" approach

  Unigrams ???

  Bigrams $\longrightarrow$ $\boxed{P(w_n|w_{n-1}) =}$

  Trigrams ???

  General case $\boxed{P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}}$

Maximum Likelihood Estimation (MLE)

Training a bigram model requires…

$$\boxed{P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}}$$

Bigram counts

Unigram counts

These are the model's *parameters*.

Today

- Recap on n-gram language models (LMs)
- Learning n-gram models: an example
- Complications when building n-gram LMs
- Part-of-speech tagging
- HMMs for sequence tagging: introduction

MLE "Raw count" bigram construction: Example with a small training dataset

<s> I get what I eat and I eat what I get </s>

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get </s>	1

<s>	1
I	4
get	2
what	2
eat	2
and	1
</s>	1

To the left is our training data:

Top: original form (training corpus).

Bottom: distillation into counts of 2-word and 1-word sequences.

“Raw count” bigram construction: An example

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

<s> I get what I eat and I eat what I get </s>

p(what | get) = **??**

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get </s>	1

<s>	1
I	4
get	2
what	2
eat	2
and	1
</s>	1

“Raw count” bigram construction: An example

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

<s> I get what I eat and I eat what I get </s>

p(what | get) = ??

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get </s>	1

<s>	1
I	4
get	2
what	2
eat	2
and	1
</s>	1

“Raw count” bigram construction: An example

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

<s> I get what I eat and I eat what I get </s>

p(what | get) = ??

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get </s>	1

<s>	1
I	4
get	2
what	2
eat	2
and	1
</s>	1

“Raw count” bigram construction: An example

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

<s> I get what I eat and I eat what I get </s>

p(what | get) = ??

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get </s>	1

<s>	1
I	4
get	2
what	2
eat	2
and	1
</s>	1

“Raw count” bigram construction: An example

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

<s> I get what I eat and I eat what I get </s>

p(what | get) = ??

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get </s>	1

<s>	1
I	4
get	2
what	2
eat	2
and	1
</s>	1

"Raw count" bigram construction: An example

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

<s> I get what I eat and I eat what I get </s>

p(what | get) = ½

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get </s>	1

<s>	1
I	4
get	2
what	2
eat	2
and	1
</s>	1

Applying the Bigram Model

<s> I get what I eat and I eat what I see .	p (<s> I get what) = p(I | <s>) x p(get | I) x p(what | get) = 1/1 x 2/4 x ½ = **0.25**

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get .	1

<s>	1
I	4
get	2
what	2
eat	2
and	1
.	1

Applying the Bigram Model

<s> I get what I eat and I eat what I see .

p (<s>I get what)
= p(I | <s>) x p(get | I) x p(what | get)
= 1/1 x 2/4 x ½ = **0.25**

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get .	1

Another note about a different sequence:

P(<s> I get what I get .) will NOT be 0, even though it isn't in the data!

The model does generalize to(some) unseen sequences.

But unseen bigrams ***will*** cause a sequence to be assigned probability 0.

<s> I get what I eat and I eat what I see .

<s> I	1
I get	2
get what	1
what I	2
I eat	2
eat and	1
and I	1
eat what	1
get .	1

Examples:

- **<s> eat and see**
- **I eat**

And that is where *smoothing* comes in…

- See notes from lecture 2…

Today

- Recap on n-gram language models (LMs)
- Learning n-gram models: an example
- Complications when building n-gram LMs
- Part-of-speech tagging
- HMMs for sequence tagging: introduction

Counting Words in Corpora

- Depends on **tokenization**
  - Should we treat punctuation marks as words
  - Important for many NLP tasks
    - Grammar-checking, spelling error detection, author identification, part-of-speech tagging
  - **Contractions**
    - Isn't vs. is n't vs. isn ' t

- **Language-dependent**
  - Freundschaftsbezeigungen = demonstration of friendship

Counting Words in Corpora

- Decisions will have an effect on performance!
- Typically, the goal is to reduce the vocabulary size. **Why?**
- And at the same time, we want to **preserve those distinctions/differences that matter** for the **downstream applications**.

Counting Words in Corpora

- Depends also on **text normalization** – string transformations that remove distinctions irrelevant to downstream applications
- **Capitalization**
  - Should *They* and *they* be treated as the same word?
    - For most statistical NLP applications, yes
    - Sometimes capitalization information is maintained as a feature
    - E.g. spelling error correction, part-of-speech tagging
- **Special fonts:** *Italics,* ***bold***
  - Usually ignore.
- **Inflected forms**
  - Should *walks* and *walk* be treated as the same word?
    - No…for most n-gram based systems

Counting Words in Corpora

- Spoken Language Corpora
  - Utterances **don't usually have punctuation**, but they do have other phenomena that we might or might not want to treat as words:

    I do uh main mainly business data processing
  - **Fragments**
  - **Filled pauses → ** *um* and *uh* behave more like words, so most speech recognition systems treat them as such

Advice: Compute everything in log space

- Avoids numerical underflow
- Adding is faster than multiplying
- Can convert back to a probability at the end if necessary by taking the exp of the logprob.

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Advice: if space is an issue

- We didn't explicitly store 0-count bigrams in the "distilled" form of the corpus into counts.
- Assume that any bigram not represented in your data structure has a count of 0.

Today

- Recap on n-gram language models (LMs)
- Learning n-gram models: an example
- Complications when building n-gram LMs
- Part-of-speech tagging
- HMMs for sequence tagging: introduction

Using n-gram models to generate language

- Assume a (trained) bigram model
  1. Generate a random bigram ***b*** that starts with <s> (according to its bigram probability).
     - Let ***w*** be the 2nd word of ***b***.
     - Output ***w***.
  2. Generate a random bigram ***b*** that starts with ***w*** (according to its bigram probability).
     - Let ***x*** be the 2nd word of ***b***.
     - Output ***x***.

  … until some stopping criterion.

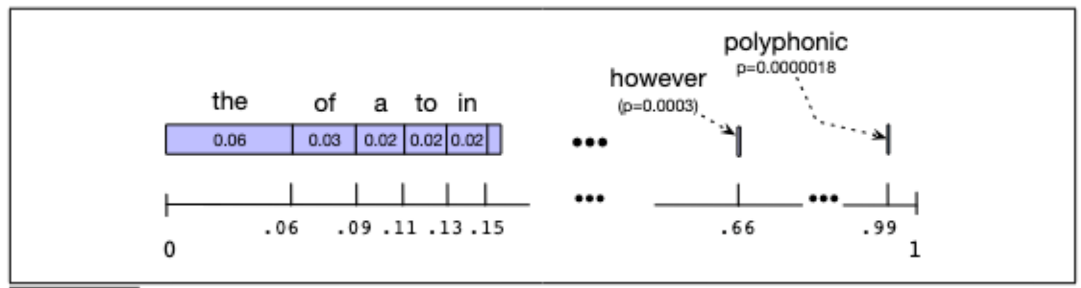
Using n-gram models to generate language

- Assume a (trained) bigram model

1. Generate a random bigram ***b*** that starts with <s> (according to its bigram probability).
   - Let *w* be the 2nd word of *b*.
   - Output *w.*
2. Generate a random bigram ***b*** that starts with *w* (according to its bigram probability).
   - Let *x* be the 2nd word of *b*.
   - Output *x.*

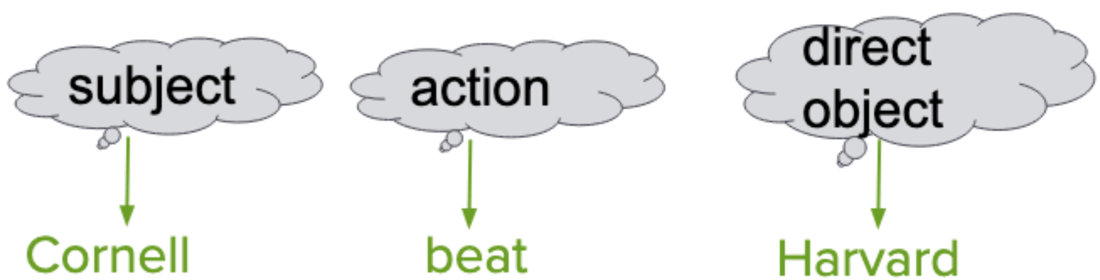
… until some stopping criterion.

Sampling (for unigrams)



People generate language with more intention

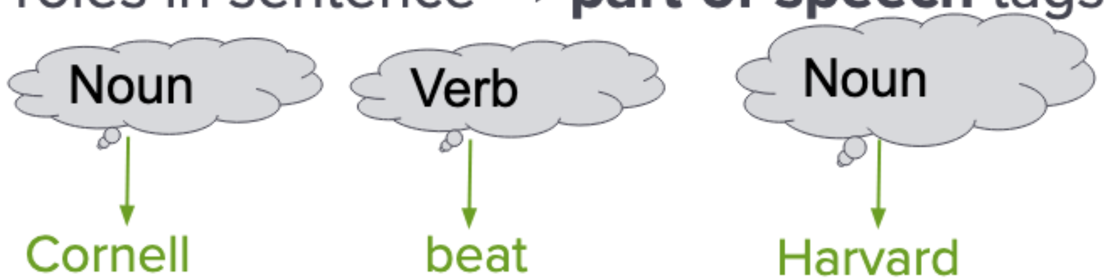
From “hidden intentions” ➜ “roles in the sentence”.



(Sentence roles actually help analyze meaning.)

A further simplification, for lecture purposes:

“roles in sentence” ➜ **part-of-speech** tags.



- We'll next look at a statistical model that adopts this notion of language generation: **Hidden Markov Model**

- In NLP it is most often used when we want to perform **sequence tagging**.

Sequence Tagging/Labeling

E.g. part-of-speech tags

Goal: Assign tags (labels) to discrete elements in a sequence

E.g. words

E.g. sentence

Part of speech tagging

"There are 10 parts of speech and they are all troublesome" – Mark Twain

- POS tags are also known as word classes, morphological classes or lexical tags
- Typically larger than Twain's 10:
  - Penn Treebank: 45
  - Brown Corpus: 87
  - C7 Target: 146
  - Universal dependency tagset: 15

Part of speech tagging

- Goal: Given a part-of-speech tagset, assign the correct part of speech tag to each word/token in a sentence

- The planet Jupiter and its moons are in effect a mini-solar system.

Part of speech tagging

- Goal: Given a part-of-speech tagset, assign the correct part of speech tag to each word/token in a sentence

- The/**DT** planet/**NN** Jupiter/**NNP** and/**CC** its/**PPS** moons/**NNS** are/**VBP** in/**IN** effect/**NN** a/**DT** mini-solar/**JJ** system/**NN** ./**.**

Part of speech tagging

- Traditionally needed as an initial processing step for a number of language technology applications:
  - Answer extraction in Question Answering systems
  - Base step in identifying syntactic phrases for IR systems
  - Critical for word sense disambiguation
  - Information extraction

What makes POS tagging hard?

- Goal: Find the correct tag for the words given the context

- Noun or Verb?

  <u>book</u> that flight

  hand me that **book**

- How do we resolve these ambiguities?

What makes POS tagging hard?

- Most word types are unambiguous wrt POS (85-86%)
- Ambiguous words account for 14-15% of the vocabulary, **BUT** they are some of the most common words of English
- Hence, 55-67% of word tokens in running text are ambiguous

		WSJ		Brown	
Types:					
Unambiguous	(1 tag)	44,432	(**86%**)	45,799	(**85%**)
Ambiguous	(2+ tags)	7,025	(**14%**)	8,050	(**15%**)
Tokens:					
Unambiguous	(1 tag)	577,421	(**45%**)	384,349	(**33%**)
Ambiguous	(2+ tags)	711,780	(**55%**)	786,646	(**67%**)

Sometimes hard for people

- Particle vs. preposition

  He talked *over* the deal

  He talked *over* the telephone
- Past tense vs. past participle

  The horse *walked* past the barn

  The horse *walked* past the barn needed more exercise
- Noun vs adjective

  The *executive* decision
- Noun vs present participle

  Fishing can be fun

Penn Treebank Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	“to”	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential ‘there’	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*’s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one’s*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

POS tagging exercise

It is a nice night

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Buffalo example

buffalo buffalo buffalo buffalo buffalo buffalo buffalo buffalo.

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo.

Buffalo buffalo, Buffalo buffalo buffalo, buffalo Buffalo buffalo.

Buffalo example

n1. the city of Buffalo, NY
n2. an animal…the American bison
v. to bully, confuse, deceive, or intimidate

Buffalo^{n1} buffalo^{n2} Buffalo^{n1} buffalo^{n2} buffalo^{v} buffalo^{v} Buffalo^{n1} buffalo^{n2}.

[Those] (Buffalo buffalo) [whom] (Buffalo buffalo) buffalo, buffalo (Buffalo buffalo).

[Those] buffalo(es) from Buffalo [that are intimidated by] buffalo(es) from Buffalo intimidate buffalo(es) from Buffalo.

Bison from Buffalo, New York, who are intimidated by other bison in their community, also happen to intimidate other bison in their community.

THE buffalo FROM Buffalo WHO ARE buffaloed BY buffalo FROM Buffalo, buffalo (verb) OTHER buffalo FROM Buffalo.

A ship shipping ship shipping shipping ships

Among easiest of NLP problems

- State-of-the-art methods achieve ~98% accuracy
- Simple heuristics can go a long way
  - ~92% accuracy just by choosing most frequent tag for word (MLE)
  - To improve reliability: ***need to use some of the local context***
- Defining rules for special cases can be time-consuming, difficult, and prone to errors / omissions
- Thus machine learning (ML) methods are employed

Today

- Recap on n-gram language models (LMs)
- Learning n-gram models: an example
- Complications when building n-gram LMs
- Part-of-speech tagging
- HMMs for sequence tagging: introduction

HMM POS Tagger

Maximizes $P(t_1 ... t_N \mid w_1 ... w_N)$

Need to Bayes flip:

$$= \frac{P(w_1 ... w_N | t_1 ... t_N) \cdot P(t_1 ... t_N)}{\cancel{P(w_1 ... w_N)}}$$

Can ignore the denominator

Problems?

Make Independence and Markov Assumptions

$P(t_1, \ldots, t_n)$: approximate using **n-gram model**

bigram $\prod_{i=1,n} P(t_i \mid t_{i-1})$

trigram $\prod_{i=1,n} P(t_i \mid t_{i-2} t_{i-1})$

Make Independence and Markov Assumptions

Assume each word appears with a particular tag independent of its neighbors

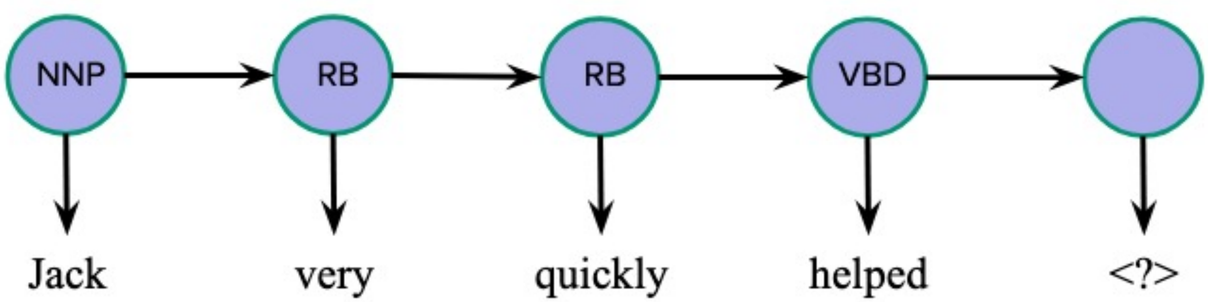
$$P(w_1 \ldots w_n \mid t_1 \ldots t_n) \cong \prod_{i=1,n} P(w_i \mid t_i)$$

After independence and Markov assumptions:

emission probabilities

$$P(t_1 \ldots t_N \mid w_1 \ldots w_N) \cong \prod_{i=1,n} P(t_i \mid t_{i-1}) \cdot P(w_i \mid t_i)$$

transition probabilities



Tagging algorithm

Given a new sentence to tag

- For every possible tag sequence,
  - Apply equation to calculate the score
- Select the highest-scoring tag sequence

Uh-oh…Too many possible tag sequences to do this!!!

Sentence length m=20

Tagset of size T = 15

$T^m = 15^{20}$ tag sequences!!!

- Stayed tuned for solution in next class…

Take-aways

1. How to construct n-gram LMs
    a. the MLE (“raw count”) method
    b. bigram LMs, and how to extend to larger n
    c. engineering tradeoffs
2. Generating text by sampling from an LM
3. Terminology: types, tokens, vocabulary/lexicon
4. Part-of-speech tagging as a sequence tagging task
5. Probabilistic model for HMM tagger

Slide Acknowledgements

- Earlier versions of this course offering including materials from Claire Cardie, Marten van Schijndel, Lillian Lee.