

Language Modeling

Professor Marie Roch

Huang et al.: 11.1-11.1.1,11.2,11.3,11.4,11.4.1

Language Models

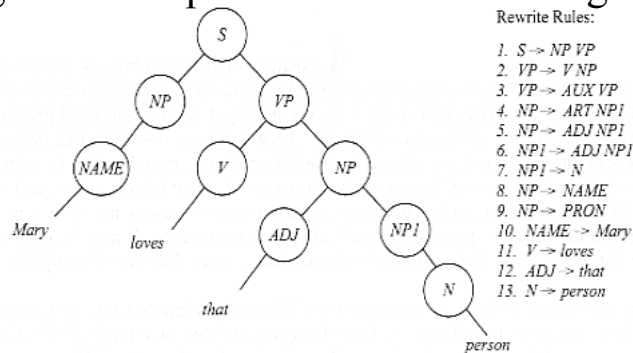
- Recall: $\Pr(\omega_i | x) = \frac{\Pr(x | \omega_i) \Pr(\omega_i)}{\Pr(x)}$
- When we consider recognizing phrases, the class ω_i is multiple words.
- Language model should constrain choices:
Pr("How may I help you?")
Pr("How dog cat walk run?")

Language Models

- Name your poison...
 - grammars
 - Well defined & understood
 - Difficult to construct & maintain
 - statistical language models
 - Simple patterns
 - Easy to construct & maintain
 - Sparse data problem – requires backoff strategies

Grammars

- A grammar is a set of rules that permit us to generate or parse sentences in a language:



Formal language theory

- A grammar is defined as $G = (V, T, P, S)$
 - V – Set of non-terminals. (Derivation is incomplete until rules are rewritten by terminals, traditionally upper case)
 - T – The set of terminals. Traditionally written in lower case.
 - P – The set of productions, or rewrite rules.
 - S – The starting terminal, a special non-terminal.

Characteristics of a grammar

- For a specific grammar, we should consider its
 - generality – What sentences does the grammar accept?
 - selectivity – What sentences does the grammar reject?
 - Understandability – How well can a human interpret and maintain the grammar?

Chomsky Hierarchy

- Language models can be partitioned into several classes which describe the generality of an abstract grammar of that class.
- Each of the classes is characterized by a grammar/automaton that corresponds to that class.

Chomsky Hierarchy

Types	Constraints	Automata
Phrase structure grammar	$\alpha \rightarrow \beta$. This is the most general grammar.	Turing machine
Context-sensitive grammar	A subset of the phrase structure grammar. $ \alpha \leq \beta $, where $ \cdot $ indicates the length of the string.	Linear bounded automata
Context-free grammar (CFG)	A subset of the context sensitive grammar. The production rule is $A \rightarrow \beta$, where A is a non-terminal. This production rule is shown to be equivalent to Chomsky normal form: $A \rightarrow w$ and $A \rightarrow BC$, where w is a terminal and B, C are non-terminals.	Push down automata
Regular grammar	A subset of the CFG. The production rule is expressed as: $A \rightarrow w$ and $A \rightarrow wB$.	Finite-state automata

Huang et al., p. 548

Note: push-down automata are also known as recursive transition networks (RTNs)

Chart Parsing

- A common data structure used for context-free grammars is the chart which we will not cover.
- You may choose to read about it in section 11.1.2 which details the chart data structure and associated parsing algorithms, but we will not be covering it in class.

Stochastic language models

- In the previous grammars, a sentence is either accepted or rejected by a grammar.
- Thus, $\Pr(\omega)$, the probability of a sequence of words is either 0 or 1. This is not particularly informative:

$$\Pr(\omega | x) = \frac{P(x | \omega)P(\omega)}{P(x)}$$

Stochastic language models

- Stochastic language models permit us to determine the probability of a word sequence.

Probabilistic context-free grammars (PCFGs)

- PCFGs assign a probability to each of the productions, i.e.:
 $VP \rightarrow V NP \quad .60$
 $VP \rightarrow AUX VP \quad .40$
- The probability of any given derivation can be determined as a function of the probabilities of the rules used.

PCFGs

- Users of PCFGs must consider that some grammars are *ambiguous*, there may be more than one parse for a sentence.
- It is unknown which parse is the correct one, and in effect, we can think of the parse tree as being hidden from us.

PCFGs

- Similar to HMMs (which we will study next) there are versions of the forward-backward and Viterbi algorithms that have been developed for use with PCFGs.
- The details are beyond the scope of this class, but are well covered in computational linguistics courses.

N-gram language models

- Let $W = (w_1, w_2, \dots, w_n)$ be a sequence of n words.

- Then

$$\begin{aligned}\Pr(W) &= \Pr(w_1 w_2 \dots w_n) \\ &= \Pr(w_1) \Pr(w_2 | w_1) \Pr(w_3 | w_1 w_2) \dots \Pr(w_n | w_1 w_2 \dots w_{n-1}) \\ &= \prod_{i=1}^n \Pr(w_i | w_1 w_2 \dots w_{i-1})\end{aligned}$$

N-grams

- Suppose that a vocabulary had v entries.
- Then for each sequence length i , there are v^i possible different sequences of words.
- The number of possible sequences grows exponentially with length, making it difficult to estimate the probability of even short sentences.

N-grams

- To make estimation tractable, we need to reduce the number of possibilities.
- One way to do this is to truncate the dependencies to the previous N-1 words by making a Markov assumption of order N-1:

$$\Pr(w_i | w_1 w_2 \dots w_{i-1}) = \Pr(w_i | w_{i-N+1} w_{i-N+2} \dots w_{i-1})$$

Special N-grams

- Unigram
 - Only depends upon the word itself.
 - $\Pr(w_i)$
- Bigram
 - $\Pr(w_i/w_{i-1})$
- Trigram
 - $\Pr(w_i/w_{i-1}, w_{i-2})$
- Quadrigram
 - $\Pr(w_i/w_{i-1}, w_{i-2}, w_{i-3})$

Bigram example

Source text: *THE CASK OF AMONTILLADO*.

THE thousand injuries of Fortunato I had borne as I best could; but when he ventured upon insult, I vowed revenge. You, who so well know the nature of my soul, will not suppose, however, that I gave utterance to a threat. At length I would be avenged ; this was a point definitively settled - but the very definitiveness with which it was resolved, precluded the idea of risk. I must not only punish, but punish with impunity. A wrong is unredressed when retribution overtakes its redresser. It is equally unredressed when the avenger fails to make himself felt as such to him who has done the wrong

Edgar Allen Poe

Prepared text

- Strip punctuation & add start/end (<s>/</s>) markers
- Possibly
 - adjust text case
 - add part of speech tags

```
<s> THE CASK OF AMONTILLADO </s> <s> THE THOUSAND INJURIES OF FORTUNATO I  
HAD BORNE AS I BEST COULD </s> <s> BUT WHEN HE VENTURED UPON INSULT I  
VOWED REVENGE </s> <s> YOU WHO SO WELL KNOW THE NATURE OF MY SOUL  
WILL NOT SUPPOSE HOWEVER THAT I GAVE UTTERANCE TO A THREAT </s>  
<s> AT LENGTH I WOULD BE AVENGED </s> <s> THIS WAS A POINT DEFINITELY  
SETTLED BUT THE VERY DEFINITIVENESS WITH WHICH IT WAS RESOLVED  
PRECLUDED THE IDEA OF RISK </s> <s> I MUST NOT ONLY PUNISH BUT PUNISH WITH  
IMPUNITY </s> <s> A WRONG IS UNREDRESSED WHEN RETRIBUTION OVERTAKES ITS  
REDRESSER </s> <s> IT IS EQUALLY UNREDRESSED WHEN THE AVENGER FAILS TO  
MAKE HIMSELF FELT AS SUCH TO HIM WHO HAS DONE THE WRONG </s>
```

Beginning & ending sentences

- Thus for the sentence:

<s> THE THOUSAND INJURIES OF FORTUNATO I HAD
BORNE AS I BEST COULD </s>

a bigram model would estimate:

Pr(THE | <s>)
Pr(THOUSAND|THE)
Pr(INJURIES|THOUSAND)
...
Pr(COULD | BEST)
Pr(</s>|COULD)

Estimating N-grams

- Estimating N-grams is done by frequency counting.
- Let $C(w_{i-N+1} w_{i-N+2} w_{i-N+3} \dots w_{i-N+N})$ denote the number of times that a sequence of N words appears in a training text.
- Sometimes, we will write this $C(w_{i-N+1}^i)$

Estimating N-grams

- Trigram example: $N=3, i=5$

$$\begin{aligned} C(w_{5-3+1}w_{5-3+2}w_{5-3+3}) &= C(w_3w_4w_5) \\ &= C(\text{CASK OF AMONTILLADO}) \end{aligned}$$

- Thus, in:

<s> THE CASK OF AMONTILLADO </s>
w1 w2 w3 w4 w5 w6

we would count the number of times that the sequence “CASK OF AMONTILLADO” occurs throughout the entire text.

Estimating N-grams

- Unigrams: trivial.
- Bigrams:
 - Count the unigrams and bigrams

$$\text{then } \Pr(w_i | w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

Estimating N-grams

- In general:

$$\Pr(w_i | w_{i-N+1} w_{i-N+2} \dots w_{i-1}) = \frac{C(w_{i-N+1}^i)}{C(w_{i-N+1}^{i-1})}$$

Sentence probability

- Determined by N-gram product.
- Bigram example:
<s> NERVOUS VERY VERY DREADFULLY
NERVOUS I HAD BEEN AND AM </s>
<s> BUT WHY WILL YOU SAY THAT I AM MAD
</s>

Sentence 1 probability = Pr(NERVOUS | <s>) Pr(VERY |
NERVOUS) Pr(VERY | VERY) Pr(DREADFULLY | VERY) Pr(I
| NERVOUS) Pr(HAD | I) ... Pr(AM | AND) Pr(</s> | AM)

(We could have derived this using the Markov property and Bayes rule.)

Sentence probability

- Other N-grams are similar.
- When using N-grams with $N > 2$, estimate the beginning of the sentence with shorter N-grams.
- Trigram example:

<s> NERVOUS VERY VERY DREADFULLY
NERVOUS I HAD BEEN AND AM </s>

= $\Pr(\text{NERVOUS} \mid \langle S \rangle) \Pr(\text{VERY} \mid \langle s \rangle \text{NERVOUS}) \Pr(\text{VERY} \mid \text{VERY NERVOUS}) \dots$

How good is our model?

- Extrinsic evaluation
 - Use it in an ASR system and test
 - Good measure
 - Expensive
- Intrinsic evaluation
 - Measure the average branching factor

Review of Entropy

- Defined as the expected amount of information (average amount of surprise).

$$\begin{aligned}H(X) &= E[I(X)] \\ &= \sum_{x_i \in S} \Pr(x_i) I(x_i) && S \text{ is all possible symbols} \\ &= \sum_{x_i \in S} \Pr(x_i) \log \frac{1}{\Pr(x_i)} && \text{definition } I(x_i) \\ &= E[-\log \Pr(X)]\end{aligned}$$

Entropy as measure of goodness

- Problem:

– Recall $H(X) = \sum_{x_i \in S} \Pr(x_i) \log \frac{1}{\Pr(x_i)}$

- distribution of X
 - real distribution unknown
 - only have estimator
- Is the estimator of any value?

Entropy as measure of goodness

- Expected value of information using estimator:

$$\hat{H}(X) = - \sum_{x_i \in S} \Pr(x_i) \log[\hat{\Pr}(x_i)]$$

- It can be shown (Jensen's \neq) that:

$$H(X) \leq \hat{H}(X)$$

$$- \sum_{x_i} \Pr(x_i) \log \Pr(x_i) \leq \sum_{x_i} - \Pr(x_i) \log \hat{\Pr}(x_i)$$

$$H(X) = \hat{H}(X) \leftrightarrow \hat{\Pr}(x_i) = \Pr(x_i)$$

Entropy as a measure of goodness

- Actual entropy is almost never known
- Estimations of language model entropy:
 - Are an upper bound on the entropy.
 - Can be used to compare two models

Perplexity

- Perplexity, defined as $PP=2^{H(X)}$, is a measure of how complex a language model is.
- The value of PP tells us how many words there would be in an equivalent dictionary where all words are equally probable.

Perplexity estimation for N-grams

- Construct the N-grams.
- As the N-grams are an approximation of the probability, we cannot compute the true entropy.
- We can however estimate entropy on a test set W with N_w words not used for training:

$$H(W) \approx -\frac{1}{N_w} \sum_{i=1}^{N_w} \log(\Pr(w_i | w_{i-N+1} w_{i-N+2} \dots w_{i-1}))$$

Perplexity example

- The Wall Street Journal corpus
 - Vocabulary size ~ 5,000
 - Bigram perplexity ~ 176
 - Trigram perplexity ~ 128
- In general, language models with lower perplexities tend to produce lower recognition error rates. (This ignores the contribution of the acoustic models.)

The need for n-gram smoothing

- Data for estimation is sparse.
- On a sample text with several million words
 - 50% of trigrams only occurred once
 - 80% of trigrams occurred less than 5 times
- Example: When pigs fly

$$\begin{aligned}\Pr(\textit{fly} | \textit{when}, \textit{pigs}) &= \frac{C(\textit{when}, \textit{pigs}, \textit{fly})}{C(\textit{when}, \textit{pigs})} \\ &= \frac{0}{C(\textit{when}, \textit{pigs})} \quad \text{if "when pigs fly" unseen}\end{aligned}$$

The need for n-gram smoothing

- If the $\Pr(\text{fly} | \text{when, pigs}) = 0$, then we will never recognize the phrase, even if somebody says it.

$$\begin{aligned}\Pr(\text{fly} | \text{when, pigs}) &= \frac{C(\text{when, pigs, fly})}{C(\text{when, pigs})} \\ &= \frac{0}{C(\text{when, pigs})} \quad \text{"when pigs fly" unseen}\end{aligned}$$

- Unfortunately, “when pigs fly” is a valid phrase that should be recognized.

Additive smoothing

- An additive smoother can simply act as if there is one more of every N-gram:

$$\begin{aligned}\Pr(\text{fly} | \text{when, pigs}) &= \frac{C(\text{when, pigs, fly}) + 1}{\sum_{w \in \text{vocab}} C(\text{when, pigs, w}) + 1} \\ &= \frac{1}{V + \sum_{w \in \text{vocab}} C(\text{when, pigs, w})} \quad \begin{array}{l} \text{"when pigs fly" unseen,} \\ V = \text{size of the vocabulary} \end{array}\end{aligned}$$

Additive smoothing

- Problem
 - What happens as our vocabulary size increases?

$$C(\text{the}, \text{white}, \text{house}) = 3$$

$$\Pr(\text{house} | \text{the}, \text{white}) = \frac{3+1}{V + \sum_{w \in \text{vocab}} C(\text{the}, \text{white}, w)}$$

- If the only other trigram starting with the white was $C(\text{the}, \text{white}, \text{horse})=2$, the original estimate would have been .60, but imagine a 10K word vocabulary...

Other N-gram smoothers

- [Deleted] Interpolation

- Backoff

Interpolation smoothing

Consider phrases:

- canine breath
- dog breath

Neither phrase in the training data.

Deleted interpolation

- Additive smoother makes the bigrams equiprobable:

$$\Pr(\text{breath}|\text{dog}) = \Pr(\text{breath}|\text{canine})$$

- Not intuitive: $P(\text{dog}) > P(\text{canine})$

Deleted interpolation

- A deleted interpolation estimate would be:

$$\Pr_I(w_i | w_{i-1}) = \lambda_{w_{i-1}} \Pr_I(w_i | w_{i-1}) + (\lambda_{w_{i-1}} - 1) \Pr(w_i)$$

- Estimation of λ_i
 - Beyond our scope
 - General idea: Use the EM algorithm and held out data

Recursive interpolation

$$\Pr_I(w_i | w_{i-2} w_{i-1}) = \lambda_{w_{i-2} w_{i-1}} \Pr_I(w_i | w_{i-2} w_{i-1}) + (\lambda_{w_{i-2} w_{i-1}} - 1) \Pr_I(w_i | w_{i-1})$$

What if $\Pr(w_i | w_{i-1}) = 0$?

$$\Pr_I(w_i | w_{i-1}) = \lambda_{w_{i-1}} \Pr_I(w_i | w_{i-1}) + (\lambda_{w_{i-1}} - 1) \Pr(w_i)$$

Consequences of deleted interpolation on a trigram model

1	The	are	to	know	the	issues	necessary
2	This	will		have	this	problems	data
3	One	the		understand	these	the	information
4	Two	would		do	problems		above
5	A	also		got	any		other
6	Three	do		the	a		time
7	Please	need		use	problem		people
8	In			provide	them		options
9	We			insert	all		tools
.				.			.
.				.			.
.				.			.
93				request			factors
94				respond			facts
95				supply			l
96				write			jobs
97				ma			MVS
98				resolve			old
.				.			.
.				.			.
.				.			.
1636				.			mailroom
1637				.			marketplace
1638				.			provision
1639				.			reception
1640				.			shop
1641				.			important

Pr(the|need to)
is probably quite
low or even absent
from the trigrams.
However, it has
high
probability in the
unigram
model.

Jelinek p. 67

Rankings for sentence: *We need to resolve all the important issues within the next two days.*

Backoff smoothing

- Basic ideas
 - Use N-gram model when possible
 - Backoff: rely on (N-1)-gram models in other case
- Problem:
 - If we use both N- and (N-1)-gram models, we no longer have a probability distribution!

Katz backoff

$$\Pr_{katz}(z | x, y) = \begin{cases} \Pr^*(z | x, y) & C(x, y, z) > 0 \\ \alpha(x, y) P_{katz}(z | y) & C(x, y) > 0, C(x, y, z) = 0 \\ \Pr^*(z) & \text{otherwise} \end{cases}$$

discounted probability (with arrow pointing to $\Pr^*(z | x, y)$)

normalizing factor (with arrow pointing to $\alpha(x, y)$)

$$\Pr_{katz}(z | y) = \begin{cases} \Pr^*(z | y) & C(y, z) > 0 \\ \alpha(y) \Pr^*(z) & \text{otherwise} \end{cases}$$

Katz backoff

- Discounted probability: \Pr^*
 - Reserve a small amount of the probability
 - Distributed to the backoff estimates α
- How much to distribute?

$$\beta(w_{n-N+1}^{n-1}) = 1 - \sum_{w_n: C(w_{n-N+1}^n) > 0} \Pr^*(w_n | w_{n-N+1}^{n-1})$$

left over probability (with arrow pointing to β)

word sequences for which we have an estimated probability (with arrow pointing to the summation term)

Katz: sharing the probability

$$\alpha(w_{n-N+1}^{n-1}) = \frac{\text{left over Pr}}{\text{Sum of Katz N-1 gram Pr's that we will use}}$$

$$= \frac{\beta(w_{n-N+1}^{n-1})}{\sum_{w_n: C(w_{n-N+1}^n)=0} P_{katz}(w_n | w_{n-N+2}^{n-1})}$$

Use n-1 gram when n gram count is 0 →

$$= \frac{1 - \sum_{w_n: C(w_{n-N+1}^n) > 0} \text{Pr}^*(w_n | w_{n-N+1}^{n-1})}{1 - \sum_{w_n: C(w_{n-N+1}^n) > 0} \text{Pr}^*(w_n | w_{n-N+2}^{n-1})}$$

everything with a zero count = 1 - everything zero count

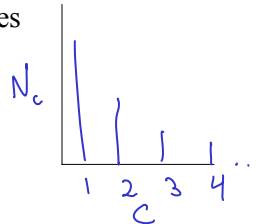
Good-Turing estimates

- Estimates discounted probability Pr^*

$$N_c = \# \text{ of } N \text{ grams that occur } c \text{ times}$$

$$= \sum_{x: C(x)=1} 1$$

- N_c 's are essentially histogram bins



- Leads to a better way to spread out the probability than adding one

Good-Turing estimates

- For each N-gram with count c , use

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

*In most cases
appears less
frequently*

- Then use c^* instead of count c for estimating Pr^*

Good-Turing: remaining probability

The left over probability is

$$\text{Pr}_{GT}^*(\text{all unseen events}) = \frac{N_1}{N}$$

which can be divided amongst the unseen N-grams

$$\text{Pr}_{GT}^*(\text{unseen event}) = \frac{1}{\#\text{unseen events}} \frac{N_1}{N}$$

Bowl with 8 types of candy

After 10 draws we have:

- 1 Snickers
- 2 Bottlecaps
- 2 Reese's
- 3 Crunch
- 1 Sweet-tarts
- 1 Kiss

No draws of

- Heath
- M&Ms

Good-Turing estimator for
Pr(Heath):

$$\Pr_{GT}^*(\text{Heath}) = \frac{1}{2} \frac{N_1}{N} = \frac{1}{2 \cdot 10} = .05$$



53

Bowl with 8 types of candy

After 10 draws we have:

- 1 Snickers
- 2 Bottlecaps
- 2 Reese's
- 3 Crunch
- 1 Sweet-tarts
- 1 Kisses

No draws of

- Heath
- M&Ms

Let's look at counts...

$$1^* = (c+1) \frac{N_{c+1}}{N_c} = 2 \frac{N_2}{N_1} = 2 \frac{2}{3} = \frac{4}{3}$$

$$2^* = 3 \frac{N_3}{N_2} = 3 \frac{1}{2} = \frac{3}{2}$$

$$\Pr_{GT}^*(\text{Snickers}) = \frac{c^*(\text{Snickers})}{N} = \frac{4}{3 \cdot 10} \approx .13$$

$$\Pr_{GT}^*(\text{Reese's}) = \frac{c^*(\text{Reese's})}{N} = \frac{3}{2 \cdot 10} = .15$$

What about Crunch? No N_4

Linear regression of N_c to c in log space. ⁵⁴

