

CS 224S LING 281 Speech Recognition and Synthesis

Lecture 11: Language Modeling
Dan Jurafsky

CS 224S W2006 1

How many words?

- I do uh main- mainly business data processing
 - Fragments
 - Filled pauses
- Are cat and cats the same word?
- Some terminology
 - Lemma**: a set of lexical forms having the same stem, major part of speech, and rough word sense
 - Cat and cats = same lemma
 - Wordform**: the full inflected surface form.
 - Cat and cats = different wordforms

CS 224S W2006 2

How many words?

- they picnicked by the pool then lay back on the grass and looked at the stars
 - 16 tokens
 - 14 types
- SWBD:
 - ~20,000 wordform types,
 - 2.4 million wordform tokens
- Brown et al (1992) large corpus
 - 583 million wordform tokens
 - 293,181 wordform types
- Let N = number of tokens, V = vocabulary = number of types
- General wisdom: $V > O(\sqrt{N})$

CS 224S W2006 3

Language Modeling

- The noisy channel model expects $P(W)$; the probability of the sentence
- The model that computes $P(W)$ is called the **language model**.
- A better term for this would be "The Grammar"
- But "Language model" or LM is standard

CS 224S W2006 4

Computing $P(W)$

- How to compute this joint probability:
 - $P(\text{"the", "other", "day", "I", "was", "walking", "along", "and", "saw", "a", "lizard"})$
- Intuition: let's rely on the Chain Rule of Probability

CS 224S W2006 5

The Chain Rule

- Recall the definition of conditional probabilities
- Rewriting:
$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$
- More generally
$$P(A \wedge B) = P(A|B)P(B)$$
- $P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$
- In general
$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \dots P(x_n|x_1 \dots x_{n-1})$$

CS 224S W2006 6

The Chain Rule Applied to joint probability of words in sentence

- $P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2) \dots P(w_n|w_1^{n-1})$
 $= \prod_{k=1}^n P(w_k|w_1^{k-1})$
- $P(\text{"the big red dog was"}) =$
- $P(\text{the}) * P(\text{big}|\text{the}) * P(\text{red}|\text{the big}) * P(\text{dog}|\text{the big red}) * P(\text{was}|\text{the big red dog})$

CS 224S W2006

Unfortunately

- Chomsky dictum: "Language is creative"
- We'll never be able to get enough data to compute the statistics for those long prefixes
- $P(\text{lizard}|\text{the,other,day,I,was,walking,along,and,saw,a})$

CS 224S W2006

Markov Assumption

- Make the simplifying assumption
 - $P(\text{lizard}|\text{the,other,day,I,was,walking,along,and,saw,a}) = P(\text{lizard}|\text{a})$
- Or maybe
 - $P(\text{lizard}|\text{the,other,day,I,was,walking,along,and,saw,a}) = P(\text{lizard}|\text{saw,a})$

CS 224S W2006

Markov Assumption

- So for each component in the product replace with the approximation (assuming a prefix of N)

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-N+1}^{n-1})$$

- Bigram version

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-1})$$

CS 224S W2006

Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

CS 224S W2006

An example

- $\langle s \rangle$ I am Sam $\langle /s \rangle$
- $\langle s \rangle$ Sam I am $\langle /s \rangle$
- $\langle s \rangle$ I do not like green eggs and ham $\langle /s \rangle$

$$P(I | \langle s \rangle) = \frac{2}{3} = .66 \quad P(\text{Sam} | \langle s \rangle) = \frac{1}{3} = .33 \quad P(\text{am} | I) = \frac{2}{2} = 1.0$$

$$P(\langle s \rangle | \text{Sam}) = \frac{1}{2} = 0.5 \quad P(\langle s \rangle | \text{Sam}) = \frac{1}{2} = 0.5 \quad P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{do} | I) = \frac{1}{1} = 1.0$$

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

- This is the Maximum Likelihood Estimate, because it is the one which maximizes $P(\text{Training set} | \text{Model})$

CS 224S W2006

More examples: Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

CS 224S W2006 14

Raw bigram counts

- Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

CS 224S W2006 14

Raw bigram probabilities

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

CS 224S W2006 15

Bigram estimates of sentence probabilities

- $P(<s> \text{ I want english food } </s>) =$
 $p(i<s>) \times p(\text{want}|i) \times p(\text{english}|want)$
 $\times p(\text{food}|english) \times p(</s>|\text{food})$
 $= .000031$

CS 224S W2006 16

What kinds of knowledge?

- $P(\text{english}|want) = .0011$
- $P(\text{chinese}|want) = .0065$
- $P(\text{to}|want) = .66$
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$
- $P(\text{want} | \text{spend}) = 0$
- $P(i | <s>) = .25$

CS 224S W2006 17

The Shannon Visualization Method

- Generate random sentences:
- Choose a random bigram $<s>$, w according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose $</s>$
- Then string the words together

```

<s> I
  I want
    want to
      to eat
        eat Chinese
          Chinese food
            Chinese food
              food </s>
    
```

CS 224S W2006 18

Unigram	<ul style="list-style-type: none"> To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have Every enter now severally so, let Hill he late speaks; or! a more to leg less first you enter Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like
Bigram	<ul style="list-style-type: none"> What means, sir. I confess she? then all sorts, he is trim, captain. Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman? Enter Menenius, if it so many good direction found 'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
Trigram	<ul style="list-style-type: none"> Sweet prince, Falstaff shall die. Harry of Monmouth's grave. This shall forbid it should be branded, if renown made it empty. Indeed the duke; and had a very good friend. Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
Quadrigram	<ul style="list-style-type: none"> King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; Will you not tell me who I am? It cannot be but so. Indeed the short and the long. Marry, 'tis a noble Lepidus.

Shakespeare as corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of $V^2= 844$ million possible bigrams: so, 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it *is* Shakespeare

CS 224B W2006

20

The wall street journal is not shakespeare (no offense)

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

CS 224B W2006

21

Lesson 1: the perils of overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
 - In real life, it often doesn't
 - We need to train robust models, adapt to test set, etc

CS 224B W2006

22

Lesson 2: zeros or not?

- Zipf's Law:
 - A small number of events occur with high frequency
 - A large number of events occur with low frequency
 - You can quickly collect statistics on the high frequency events
 - You might have to wait an arbitrarily long time to get valid statistics on low frequency events
- Result:
 - Our estimates are sparse! no counts at all for the vast bulk of things we want to estimate!
 - Some of the zeroes in the table are really zeros. But others are simply low frequency events you haven't seen yet. After all, ANYTHING CAN HAPPEN!
 - How to address?
- Answer:
 - Estimate the likelihood of unseen N-grams!

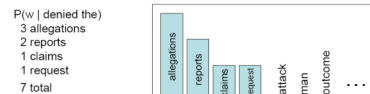
Slide adapted from Bonnie Hout and Julia Hirschberg

CS 224B W2006

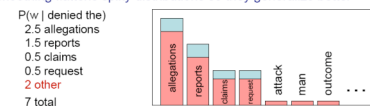
23

Smoothing is like Robin Hood: Steal from the rich and give to the poor (in probability mass)

- We often want to make predictions from sparse statistics:



- Smoothing flattens spiky distributions so they generalize better



- Very important all over NLP, but easy to do badly!

Slide from Ryan Shiro

CS 224B W2006

24

Add-one smoothing

- Also called Laplace smoothing
- Just add one to all the counts!
- Very simple

MLE estimate: $P(w_i) = \frac{c_i}{N}$

Laplace estimate: $P_{\text{addone}}(w_i) = \frac{c_i + 1}{N + V}$

Reconstructed counts: $c_i^* = (c_i + 1) \frac{N}{N + V}$

CS 224S W2006 24

Add-one smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

CS 224S W2006 26

Add-one bigrams

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

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.0062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

CS 224S W2006 27

Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

	want	to	eat	chinese	food	lunch	spend	
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

CS 224S W2006 28

Note big change to counts

- C(count to) went from 608 to 238!
- P(to|want) from .66 to .26!
- Discount $d = c^*/c$
 - d for "chinese food" = .10!!! A 10x reduction
 - So in general, add-one is a blunt instrument
 - Could use more fine-grained method (add-k)
- But add-one smoothing not used for N-grams, as we have much better methods
- Despite its flaws it is however still used to smooth other probabilistic models in NLP, especially
 - For pilot studies
 - in domains where the number of zeros isn't so huge.

CS 224S W2006 29

Better discounting algorithms

- Intuition used by many smoothing algorithms
 - Good-Turing
 - Kneser-Ney
 - Witten-Bell
- Is to use the count of things we've seen once to help estimate the count of things we've never seen

CS 224S W2006 30

Good-Turing: Josh Goodman intuition

- Imagine you are fishing
- You have caught
 - 10 carp, 3 cod, 2 tuna, 1 trout, 1 salmon, 1 eel
- How likely is it that next species is new?
 - 3/18
- Assuming so, how likely is it that next species is tuna?
 - Less than 2/18

Slide from Josh Goodman

CS 224S W2006

31

Good-Turing Intuition

- Notation: N_x is the frequency-of-frequency- x
 - So $N_{10}=1$, $N_1=3$, etc
- To estimate total number of unseen species
 - Use number of species (words) we've seen once
 - $c_0^* = c_1$ $p_0 = N_1/N$
- All other estimates are adjusted (down) to give probabilities for unseen

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

Slide from Josh Goodman

CS 224S W2006

32

Good-Turing Intuition

- Notation: N_x is the frequency-of-frequency- x
 - So $N_{10}=1$, $N_1=3$, etc
- To estimate total number of unseen species
 - Use number of species (words) we've seen once
 - $c_0^* = c_1$ $p_0 = N_1/N$ $p_0 = N_1/N = 3/18$
- All other estimates are adjusted (down) to give probabilities for unseen

$$c^* = (c + 1) \frac{N_{c+1}}{N_c} \quad P(\text{eel}) = c^*(1) = (1+1) 1/3 = 2/3$$

Slide from Josh Goodman

CS 224S W2006

33

Bigram frequencies of frequencies and GT re-estimates

AP Newswire			Berkeley Restaurant—		
c (MLE)	N_c	c^* (GT)	c (MLE)	N_c	c^* (GT)
0	74,671,100,000	0.0000270	0	2,081,496	0.002553
1	2,018,046	0.446	1	5315	0.533960
2	449,721	1.26	2	1419	1.357294
3	188,933	2.24	3	642	2.373832
4	105,668	3.24	4	381	4.081365
5	68,379	4.22	5	311	3.781350
6	48,190	5.19	6	196	4.500000

CS 224S W2006

34

Complications

- In practice, assume large counts ($c > k$ for some k) are reliable:

$$c^* = c \text{ for } c > k$$

- That complicates c^* , making it:

$$c^* = \frac{(c + 1) \frac{N_{c+1}}{N_c} - c \frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}, \text{ for } 1 \leq c \leq k.$$

CS 224S W2006

35

Backoff and Interpolation

- Another really useful source of knowledge
- If we are estimating:
 - trigram $p(z|xy)$
 - but $c(xyz)$ is zero
- Use info from:
 - Bigram $p(z|y)$
- Or even:
 - Unigram $p(z)$
- How to combine the trigram/bigram/unigram info?

CS 224S W2006

36

Backoff versus interpolation

- Backoff: use trigram if you have it, otherwise bigram, otherwise unigram
- Interpolation: mix all three

CS 224S W2006

37

Katz Backoff

$$P_{\text{katz}}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{if } C(w_{n-N+1}^{n-1}) > 0 \\ \alpha(w_{n-N+1}^{n-1})P_{\text{katz}}(w_n|w_{n-N+2}^{n-1}), & \text{otherwise.} \end{cases}$$

$$P_{\text{katz}}(w_j|w_{j-2}w_{j-1}) = \begin{cases} P^*(w_j|w_{j-2}w_{j-1}), & \text{if } C(w_{j-2}w_{j-1}) > 0 \\ \alpha(w_{j-1}w_j)P^*(w_j|w_{j-1}), & \text{else if } C(w_{j-1}w_j) > 0 \\ \alpha(w_j)P^*(w_j), & \text{otherwise.} \end{cases}$$

CS 224S W2006

38

GT smoothed bigram probs

	i	want	to	eat	chinese	food	lunch	spend
i	0.0014	0.326	0.00248	0.00355	0.000205	0.0017	0.00073	0.000489
want	0.00134	0.00152	0.656	0.000483	0.00455	0.00455	0.00384	0.000483
to	0.000512	0.00152	0.00165	0.284	0.000512	0.0017	0.00175	0.0873
eat	0.00101	0.00152	0.00166	0.00189	0.0214	0.00166	0.0563	0.000585
chinese	0.00283	0.00152	0.00248	0.00189	0.000205	0.519	0.00283	0.000585
food	0.0137	0.00152	0.0137	0.00189	0.000409	0.00366	0.00073	0.000585
lunch	0.00363	0.00152	0.00248	0.00189	0.000205	0.00131	0.00073	0.000585
spend	0.00161	0.00152	0.00161	0.00189	0.000205	0.0017	0.00073	0.000585

CS 224S W2006

39

Intuition of backoff+discounting

- How much probability to assign to all the zero trigrams?
 - Use GT or other discounting algorithm to tell us
- How to divide that probability mass among different contexts?
 - Use the N-1 gram estimates to tell us
- What do we do for the unigram words not seen in training?
 - Out Of Vocabulary** = OOV words

CS 224S W2006

40

OOV words: <UNK> word

- Out Of Vocabulary** = OOV words
- We don't use GT smoothing for these
 - Because GT assumes we know the number of unseen events
- Instead: create an unknown word token <UNK>
 - Training of <UNK> probabilities
 - Create a fixed lexicon L of size V
 - At text normalization phase, any training word not in L changed to <UNK>
 - Now we train its probabilities like a normal word
 - At decoding time
 - If text input: Use UNK probabilities for any word not in training

CS 224S W2006

41

Practical Issues

- We do everything in log space
 - Avoid underflow
 - (also adding is faster than multiplying)

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

CS 224S W2006

42

ARPA format

unigram: $\log p^*(w_i)$ w_i $\log \alpha(w_i)$
 bigram: $\log p^*(w_i|w_{i-1})$ $w_{i-1}w_i$ $\log \alpha(w_{i-1}w_i)$
 trigram: $\log p^*(w_i|w_{i-2},w_{i-1})$ $w_{i-2}w_{i-1}w_i$

```
\data\
ngram 1=1447
ngram 2=9420
ngram 3=5201

\1-grams:
-0.8679678 </a>
-99 <a>
-4.743076 chow-fun
-4.266155 fries
-3.175167 thursday
-1.776296 want
...

\2-grams:
-0.6077676 <a> i
-0.4861297 i want
-2.832415 to drink
-0.5469525 to eat
-0.09403705 today </s>
...

\3-grams:
-2.579416 <a> i prefer
-1.148009 <a> about fifteen
-0.4120701 to go
-0.3735807 me a list
-0.260361 at jupiter </s>
-0.260361 a malaysian restaurant
...
\end\
```

Language Modeling Toolkits

- SRILM
- CMU-Cambridge LM Toolkit

Instead of backoff: Interpolation

- Simple interpolation

$$\hat{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1 P(w_n|w_{n-1}w_{n-2}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

- Lambda's conditional on context:

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 (w_{n-2}^{\frac{1}{2}}) P(w_n|w_{n-2}w_{n-1}) + \lambda_2 (w_{n-2}^{\frac{1}{2}}) P(w_n|w_{n-1}) + \lambda_3 (w_{n-2}^{\frac{1}{2}}) P(w_n)$$

Evaluating N-gram models

- Best evaluation for an N-gram
 - Put model A in a speech recognizer
 - Run recognition, get WER for A
 - Put model B in speech recognition, get WER for B
 - Compare WER for A and B
- But
 - This is really time-consuming
 - Can take days to run an experiment
- So
 - As a temporary solution, in order to run experiments
 - To evaluate N-grams we often use a (poor) approximation called **perplexity**
 - But perplexity is a poor approximation unless the test data looks **just** like the training data
 - So is **generally only useful in pilot experiments (generally is not sufficient to publish)**

Perplexity Intuition

- How hard is the task of recognizing digits '0,1,2,3,4,5,6,7,8,9,oh': easy, perplexity 11
- How hard is recognizing (30,000) names at Microsoft. Hard: perplexity = 30,000
- If a system has to recognize
 - Operator (1 in 4)
 - Sales (1 in 4)
 - Technical Support (1 in 4)
 - 30,000 names (1 in 120,000 each)
 - Perplexity is 54
- Perplexity is weighted equivalent branching factor

The math

- Perplexity is technically
 - “the exp of the cross-entropy of the model on the test data”
 - So let’s review some information theory

CS 224S W2006 49

- The entropy of a random variable X :

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \log_2 p(x)$$

- Cover and Thomas example:
 - Sending a (now defunct) Western Union telegram about which horse to bet on, of 8 horses
 - Could just encode horse number with 3 binary bits
 - So on average, if we bet all day, we’ll be sending:
 - 3 bits/bet to specify the horse
 - Can we do better?

CS 224S W2006 50

We could use less bits!

- Suppose the distribution over horses we’re betting on:

Horse 1	$\frac{1}{2}$	Horse 5	$\frac{1}{64}$
Horse 2	$\frac{1}{4}$	Horse 6	$\frac{1}{64}$
Horse 3	$\frac{1}{8}$	Horse 7	$\frac{1}{64}$
Horse 4	$\frac{1}{16}$	Horse 8	$\frac{1}{64}$

- Then the entropy = best code would require:

$$\begin{aligned} H(X) &= - \sum_{i=1}^{i=8} p(i) \log p(i) \\ &= -\frac{1}{2} \log \frac{1}{2} - \frac{1}{4} \log \frac{1}{4} - \frac{1}{8} \log \frac{1}{8} - \frac{1}{16} \log \frac{1}{16} - 4 \left(\frac{1}{64} \log \frac{1}{64} \right) \\ &= 2 \text{ bits} \end{aligned}$$

CS 224S W2006 51

But we need to compute entropy over sequences

- Entropy of a sequence

$$H(w_1, w_2, \dots, w_n) = - \sum_{W_1^n \in \mathcal{L}} p(W_1^n) \log p(W_1^n)$$

- Entropy rate (per-word entropy)

$$\frac{1}{n} H(W_1^n) = - \frac{1}{n} \sum_{W_1^n \in \mathcal{L}} p(W_1^n) \log p(W_1^n)$$

CS 224S W2006 52

But to measure true entropy of a language

- We need to think about sequences of infinite length
- Consider language as a stochastic process L that generates words:

$$\begin{aligned} H(L) &= \lim_{n \rightarrow \infty} \frac{1}{n} H(w_1, w_2, \dots, w_n) \\ &= \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{W \in \mathcal{L}} p(w_1, \dots, w_n) \log p(w_1, \dots, w_n) \end{aligned}$$

- Shannon-McMillan-Breiman Theorem:

$$H(L) = \lim_{n \rightarrow \infty} - \frac{1}{n} \log p(w_1 w_2 \dots w_n)$$

CS 224S W2006 53

Cross-Entropy

- Given a distribution p , and our attempt to model it m ,
- The cross entropy of m on p is:

$$H(p, m) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{W \in \mathcal{L}} p(w_1, \dots, w_n) \log m(w_1, \dots, w_n)$$

- Draw sequences according to p , sum probabilities according to m
- Shannon-McM-Breiman theorem version:

$$H(p, m) = \lim_{n \rightarrow \infty} - \frac{1}{n} \log m(w_1 w_2 \dots w_n)$$

CS 224S W2006 54

Cross-entropy

- For any model m

$$H(p) \leq H(p, m)$$

- I.e., cross-entropy is an upper bound on entropy
- And the better m is
- The closer the cross-entropy is to the true entropy $H(p)$
- So we can use cross-entropy to compare models
- The better model has the lower cross-entropy

CS 224S W2006 44

Finally: perplexity

- SMB: if we have a model $M = P(W)$:

$$H(W) = -\frac{1}{N} \log P(w_1 w_2 \dots w_N)$$

$$\begin{aligned} \text{Perplexity}(W) &= 2^{H(W)} \\ &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}} \\ &= \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}} \end{aligned}$$

CS 224S W2006 46

- Minimizing perplexity = maximizing probability of test set according to model

$$\begin{aligned} \text{Perplexity}(W) &= 2^{H(W)} \\ &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}} \\ &= \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}} \end{aligned}$$

- Example:
 - Perplexity of N-gram:
 - $V = 20K$
 - Training = 38M words
 - Test = 1.5 M words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

CS 224S W2006 47

Advanced LM stuff

- Current best smoothing algorithm
 - Kneser-Ney smoothing
- Other stuff
 - Variable-length n-grams
 - Class-based n-grams
 - Clustering
 - Hand-built classes
 - Cache LMs
 - Topic-based LMs
 - Sentence mixture models
 - Skipping LMs
 - Parser-based LMs

CS 224S W2006 48

Summary

- LM
 - N-grams
 - Discounting: Good-Turing
 - Katz backoff with Good-Turing discounting
 - Interpolation
 - Unknown words
 - Evaluation:
 - Entropy, Entropy Rate, Cross Entropy
 - Perplexity
 - Advanced LM algorithms

CS 224S W2006 49