Natural Language Processing

Lecture #3
Morphological Analysis PartII

Hutchatai Chanlekha

Department of Computer Engineering, Kasetsart University

Can you guess the missing word?

- There are seven ……. in a week.
- He ……. on the front door.
- Can you ….. that sound?
- ฝน….หนักมาก
- เธอนั่งพับเพียบ….มาลัย
- เขา….น้ำต้นไม้ทุกเช้า

How can you make computer guess the missing word?

Guessing Next Word

- Word prediction is an essential subtask for
  - Speech recognition
  - Hand-writing recognition
  - Augmentative communication for the disabled
  - Spelling error detection
  - Etc.
- Looking at previous words can give us an important cue about what the next ones are going to be.
  - “I have a …”, the thief said.
  - “I have a sun”, the thief said.
  - “หิว…”
  - “หิวบ่าว”

“I have a gun,” the thief said

หิวข้าว

Guessing next word: Spelling error

- Example:
  - "They are leaving in about fifteen *minuets* to go to her house."
  - "The study was conducted mainly *be* John Black."
  - "Can you *lave* him my messages?"
  - "He is trying to *fine* out."
- Spell checkers can look for low probability combinations
  - Caution: Sentences with no spelling errors may also have low probability word sequences
- Algorithms that make use of surrounding words and other features to do context-sensitive spelling error correction.

Collocation

- *Collocation* is defined as a sequence of words or terms which co-occur more often than would be expected by chance.

- Compositional
  - The meaning of the expression can be predicted from the meaning of its parts.

- Most collocations exhibit milder forms of non-compositionality
  - Usually an element of meaning added to the combination.
    - Strong tea -: strong = “rich in some active agent” instead of “having great physical strength”
  - Idioms are the most extreme examples of non-compositionality

Collocation (cont.)

Frequency	Word 1	Word 2
80871	of	the
58841	in	the
26430	to	the
21842	on	the
21839	for	the
18568	and	the
16121	that	the
15630	at	the
15494	to	be
13899	in	a
13689	of	a
13361	by	the
13183	with	the
12622	from	the
11428	New	York
10007	he	said
9775	as	a
9231	is	a
8753	has	been
8573	for	a

- Finding collocation
  - Only frequency doesn't work
  - Consider only frequency results in many useless pairs
- Justeson and Katz (1995) use simple heuristic to improves these results
  - Pass the candidate phrases through a part-of-speech filter → patterns such as A N, N N, A A N, etc.
  - This approach combined frequency filter with a small amount of linguistic knowledge

Collocation (cont.)

- What if collocations consist of two words that stand in a more flexible relationship to one another?
  - She knocked on his door.
  - They knocked at the door.
  - 100 women knocked on Donaldson's door.
  - A man knocked on the metal front door.
- Fixed phrase approach will not work for such case.
- How to handle this problem??

Collocation (cont.)

- To solve the problem: use collocation window
- Window size: usually 3 to 4 words on each side of a word
- Consider every word pair as a collocational bigram, then calculate frequency

Sentence: Stocks crash as rescue plan teeters

Bigram:	Stocks crash	Stocks as crash as	Stocks rescue crash rescue as rescue	crash plan as plan rescue plan	as teeters rescue teeters plan teeters

The strategy for listing all the candidates for collocation

Discovering Collocation: Mean & Variance

- One way to discover the relationship between knocked and door
  - Compute 'mean' and 'variance' of the offsets between the two words in the corpus
  - If 'door' occurs before 'knock', the offset is a negative number
    - Mean = average offset
    - ¼* (3+3+5+5) = 4.0

She knocked on his door.
They knocked at the door.
100 women knocked on Mr. Donaldson's door.
A man knocked on the metal front door.

  - Variance measures how much the individual offsets deviate from the mean.

$$S^2 = \frac{\sum_{i=1}^{n}(d_i - \bar{d})^2}{n-1}$$

Deviation of the pair knocked/door from the example

$$= \sqrt{\frac{1}{3}((3-4.0)^2 + (3-4.0)^2 + (5-4.0)^2 + (5-4.0)^2)} = 1.15$$

  - Sample deviation; S

$$S = \sqrt{S^2}$$

- Mean and deviation characterize the distribution of distances between two words in corpus.
  Low deviation means the two words usually occur at about the same distance

Discovering Collocation: MI

- Pointwise Mutual Information

$$I(x', y') = \log_2 \frac{P(x', y')}{P(x')P(y')}$$

$$= \log_2 \frac{P(x' \mid y')}{P(x')}$$

$$= \log_2 \frac{P(y' \mid x')}{P(y')}$$

  - Roughly a measure of how much one word tell us about the other
  - Mutual information is a good measure of independence
    - MI close to 0 indicate independence
  - Not a good measure for dependence
    - For dependence, the score depends on the frequency of the individual words
    - Bigrams of two low freq. words will receive a higher score than bigrams of high freq. words

Discovering Collocation: Hypothesis Testing

- Hypothesis Testing
  - High frequency and low variance can be accidental
    - Test whether two words occur together more often than chance
    - Hypothesis testing
      - Null hypothesis; H_0
        - There is no association between the words beyond chance occurrences
      - Compute probability *p* that the event would occur if H_0 were true.
      - Reject H_0 if *p* is too low ← collocation is not by chance
        - such as $p < 0.05$, $p < 0.01$, $p < 0.005$, $p < 0.0001$
      - Otherwise retain H_0

Hypothesis Testing

- Formulating null hypothesis
  - H_0: should be true if two words do not form a collocation
    - w^1, w^2 is generated completely independently of the other
    - So, chance of appearing together is

$$P(w^1 w^2) = P(w^1)P(w^2)$$

Hypothesis Testing: *t* test

- $$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}}$$
  - If *t* is large enough, reject H_0

$\bar{x}$	: observed mean (sample mean)
μ	: expected mean (mean of the distribution)
S^2	: sample variance
N	: sample size

- Look up the value of *t* that corresponds to a significant level of α = C (e.g. C=0.005) in the lookup table
- If computed *t* is larger than C → reject H_0
- *t*-test takes into account the number of co-occurrences of the bigram relative to the frequencies of the component words

Example of *t* test

- Example: new companies
  - Suppose: corpus size = 14,307,668
    - |N| = 14,307,668
    - '*new*' occurs 15828 times; 'companies' occurs 4675
    - '*new companies*' occurs 8 times
  - $H_0 = P(new\ companies) = P(new)P(companies) = \frac{15828}{14307668} * \frac{4675}{14307668} = 3.615*10^{-7}$
    - Computed based on the assumption that the two words are independent

μ

  - Calculating *t*

$$\bar{x} = {}^{8}\!/_{14307668} \approx 5.591*10^{-7}$$

$$S^2 = p(1-p) \approx p \quad = 5.59110*10^{-7}$$

  - $t = 0.99932 \quad < 2.576$
    - *Can't reject* H_0
    - → do not form collocation

	p C	0.05 90%	0.025 95%	0.01 98%	0.005 99%	0.001 99.8%	0.0005 99.9%
d.f	1	6.314	12.71	31.82	63.66	318.3	636.6
	10	1.812	2.228	2.764	3.169	4.144	4.587
	20	1.725	2.086	2.528	2.845	3.552	3.580
(z)	∞	1.645	1.960	2.326	**2.576**	3.091	3.291

Hypothesis Testing: chi-square test (1)

- *t*-test assumes normal distribution of probability
- χ^2 does not assume normally distributed probability
  - Compare observed frequencies with frequencies expected for independent
  - If the difference is large, then reject H_0
- (1) $$\chi^2 = \sum_{i,j} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

  i → row; j → column;

  O_{ij} → observed value for cell i,j

  E_{ij} → expected value for cell i,j
- If we assume two words occur independently
  - E_{ij} = P('w_1' occur as the 1st part of bigram) * P('w_2' occur as the 2nd part of bigram) * N

any bigram

- Equation (1), can be derived as:

$$\chi^2 = \frac{N(O_{11}O_{12} - O_{21}O_{22})^2}{(O_{11}+O_{12})(O_{11}+O_{21})(O_{12}+O_{22})(O_{21}+O_{22})}$$

	$W_1 = X$	$W_1 \neq X$
$W_2 = Y$	n1 (X Y)	n2 (e.g. Q Y)
$W_2 \neq Y$	n3 (e.g. X R)	n4 (e.g. Q R)

Hypothesis Testing: chi-square test (2)

- Suppose
  - 14307668 tokens
  - C(new) = 15828, C(companies) = 4675, C(new companies) = 8
- If we assume two words occur independently
  - E_{ij} = P('*new*' occur as 1st part of bigram) * P('*companies*' occur as 2nd part of bigram) * N
    = ((8+15820) / N) * ((8+4667) / N) * N = 5.2
- From

$$\chi^2 = \frac{N(O_{11}O_{12} - O_{21}O_{22})^2}{(O_{11}+O_{12})(O_{11}+O_{21})(O_{12}+O_{22})(O_{21}+O_{22})}$$

	W_1 = new	$W_1 \neq$ new
W_2 = companies	8	4667
$W_2 \neq$ companies	15820	14287178

$\chi^2 = 1.53$

at d.f. = 1 : $\alpha = 0.05$ → $\chi^2 = 3.841$
So, we cannot reject H_0, i.e. '*new*' and '*companies*' occur independently

Other Hypothesis Testing

- Likelihood ratios
- Relative frequency ratios
- Etc.

Let's think ....

- *"I have a work to do" "go you must tomorrow"*
- What is the probability of this string of words?

- *"Do you want to _____"*
- How can we guess the next word?

- *"More heavy rain is ____ for this area in the coming days"*
  a) predicted b) believed c) forecasted d) seen
- How can we choose the most probable next-word?

N-gram Model

- Ways to assign probabilities to strings of words
  - Computing probability of an entire sentence
  - Giving a probabilistic prediction of what the next word will be in a sequence.
- For calculating probability, it is better to look at conditional probability of a word given previous words instead of individual relative frequencies of words.
  - **N-gram**: One of the models for word prediction
- How to compute the probability of a complete string of words

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

$$= \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

N-gram model (3)

- Past behavior is a good guide to what will happen in the future.

$$P(w_n \mid w_1 \ldots w_{n-1})$$ ← Predict next word base on previous words

  - n=2: Bigram
    - $P(w_2 \mid w_1)$
  - n=3: Trigram
    - $P(w_3 \mid w_1 w_2)$
  - n=4: Four-gram
    - $P(w_4 \mid w_1 w_2 w_3)$

	2-Gram	3-Gram	4-Gram	5-Gram	6-Gram	7-Gram
เจ้าหน้าที่	ตำรวจ	รับ	แจ้ง	เหตุ	ทะเลาะ	วิวาท
ทีม	ไทย	ผ่าน	เข้า	รอบ	รอง	ชนะเลิศ
พบ	ชาย	ถูก	แทง	ได้รับ	บาดเจ็บ	สาหัส

N-gram model (4)

- In principle, we want *n* in n-gram model to be fairly large.

She swallowed a large green ……

  - *Swallow* is quite strongly influencing which word will come next.
    - cake, pill? √ car, hill, tree? ✗
- But!!
  - Too large *n* → sparseness
    - Large *n* → too many parameters to estimate
  - Suppose vocabulary size = 20,000
    - 2-gram → 20000^2 ≈ 400 million
    - 3-gram → 20000^3 ≈ 8 trillion
    - 4-gram → 20000^4 ≈ $1.6*10^{17}$
  - So, *n*-gram system usually use bigram or trigram

N-gram model (5)

- Using large n-gram is impractical
- Simplification: approximate the probability of a word given all the previous words.
  - bigram :- the probability of the word given the single previous word.
  - P(rabbit | Just the other day I saw a) = P(rabbit | a)
- Markov assumption
  - Only prior local context (i.e. the last few words) affects the next word.
  - Bigram : first-order Markov model
  - Trigram: second-order Markov model
  - N-gram: *(N-1)th*-order Markov model
- By using bigram as approximation

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

N-gram model (6)

- N-gram models work best when trained on enormous amounts of data
  - N-gram based systems usually process sentence-by-sentence
    - In this case, start-of-a-sentence does not depend on last words of the preceding sentence, but depend on a dummy 'Beginning-of-sentence'.

- N-gram

$$P(w_n \mid w_1 \ldots w_{n-1}) = \frac{P(w_1 \ldots w_n)}{P(w_1 \ldots w_{n-1})}$$

- How can we estimate next word?

Next-word Probability Estimation

- Suppose we use tri-gram model for estimating next word.
  - Use two preceding words to estimate next words
    - “come” “across”
  - If we found 10 training instances of he two consecutive words ‘come across’
    - 8 of 10 followed by ‘as’
    - 1 of 10 followed by ‘more’
    - 1 of 10 followed by ‘a’
  - What probability estimates we should use for estimating next words?

- Relative frequency as a probability estimator
  - P(as) = 0.8 P(more) = 0.1 P(a) = 0.1
  - P(x) = 0; for x not among the above 3 words

Example

- Suppose we want to compute the probability of the sentence:
  "I want to eat British food"

<S> I = .25	I want = .32	want to = .65	to eat = .26	British food = .60
<S> I'd = .60	I would = .29	want a = .14	to have = .14	British restaurant = .15
<S> Tell = .04	I don't' = .08	want some = .04	to spend = .09	British cuisine = .01
<S> I'm = .02	I have = .04	want thai = .01	to be = .02	British lunch = .01

- P(I want to eat British food) = P(I|<S>) * P(want|I) * P(to|want) * P(eat|to) *
  P(British|eat) * P(food|British)
  = 0.25*0.32*0.65*0.26*0.002*0.60 = .000016
- Problem!!
  - Multiplication of many probabilitisties ➔ risk of numerical underflow
  - More customary to do computation in log space

* Bigram from Berkeley Restaurant Project.

Calculating n-gram

- Counting and normalizing
- Technique known as Maximum Likelihood Estimation (MLE)
  - Maximum likelihood estimate (ML)

    $P_{MLE}(w_1 \ldots w_n) = C(w_1 \ldots w_n) / N$;N = number of training instance

    $P_{MLE}(w_n | w_1 \ldots w_{n-1}) = C(w_1 \ldots w_n) / C(w_1 \ldots w_{n-1})$

    - $C(w_1 \ldots w_n)$ → frequency of n-gram $w_1 \ldots w_n$ in training text
- Example: bigram

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

All bigram counts that start with w_{n-1} is equal to unigram count

- For general case of N-gram parameter estimation

$$P(w_n \mid w_{n-N+1}^{n-1}) = C(w_{n-N+1}^{n-1} w_n) \Big/ C(w_{n-N+1}^{n-1})$$

Estimate N-gram prob. By dividing observed freq. of a particular sequence by observed freq. of a prefix

More on N-grams and Sensitivity to Training Corpus

- Increasing value of N ➔ Increasing the accuracy of N-gram models

- Unigram
  - Every enter now severally so, let
  - Are where extent and sighs have rise excellency took of .. Sleep knave we. Near; vile like
- Bigram
  - What means, sir. I confess she? Then all sorts, he is trim, captain.
  - What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentelman?
- Trigram
  - What isn't that cried?
  - Sweet prince, Falstaff shall die. Harry of Monmouth's grave
- 4-gram
  - Will you not tell me who I am?
  - They say all lovers swear more performance than they are wont to keep obliged faith unforfeited !

* From corpus of Shakespeare's works

More on N-grams and Sensitivity to Training Corpus

- Strong dependency on the training corpus
  - N-gram generated from different corpus (i.e. genre, domain, etc.)

- 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

* Generated from 40 million words of articles from Wall Street Journal

Problem with N-gram

- MLE is general unsuitable for statistical inference in NLP
  - Because of the sparseness of the data
    - Few words are common, vast majority of words are very uncommon
    - Longer n-gram involving rare words are much rarer
  - MLE assign zero probability to unseen events
    - *Prob.* of a long string is generally computed by multiplying *Prob.* of subparts
    - Zero will propagate and result in bad estimates for *prob.* of sentences when the certain *n*-grams never occurred in training text.
  - Does it help if help we collect much more data?
    - Probably not!!!
    - Devise better estimators
      - Discounting method (smoothing)
        - Allow for possibility that we might see events that we didn't see in training corpus

Add-One Smoothing

- Add 1 to all the N-gram counts before normalize
- Simple, does not perform well, is not commonly used

- Unigram probability:

  $P(w_x) = C(w_x)/N$;N = total number of word tokens

- Add-one smoothing

$$p_i^* = \frac{c_i + 1}{N + V}$$

Adding 1 to the count for each word type, the total number of tokens must be increased by the number of types; V (total number of word types in the language)

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

Normalization factor

  - For bigram

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

Example of Add-One smoothing

- From Berkeley Restaurant Project's corpus (with 1616 total word types, N=3367)

	I	want	to	eat
I	8	1087	0	13
want	3	0	786	0
to	3	0	10	860
eat	0	0	2	0

Bigram counts

	I	want	to	eat
I	.0023	.32	0	.0038
want	.0025	0	.65	0
to	.00092	0	.0031	.26
eat	0	0	.0021	0

Bigram probabilities

	I	want	to	eat
I	9	1088	1	14
want	4	1	787	1
to	4	1	11	861
eat	1	1	3	1

Add-one Smoothed Bigram counts

	I	want	to	eat
I	.0018	.22	.0002	.0028
want	.0014	.00035	.28	.00035
to	.00082	.00021	.0023	.18
eat	.00039	.00039	.0012	.00039

Add-one Smoothed Bigram probabilities

Example of Add-One smoothing (cont.)

- From table Add-one Smoothed Bigram prob., lets reconstruct count matrix

  from $p^*(w_n|w_{n-1})=(C(w_{n-1}w_n)+1)/(C(w_{n-1})+V)$

	I	want	to	eat
I	8	1087	0	13
want	3	0	786	0
to	3	0	10	860
eat	0	0	2	0

Original counts

	I	want	to	eat
I	6	740	.68	10
want	2	.42	331	.42
to	3	.69	8	594
eat	.37	.37	1	.37

Reconstruct count

- From reconstructed table
  - Add-one smoothing made a very big change in counts
  - Probability space also decreases
  - Discount d_c is high!!
  - Sharp change in counts and probabilities occurs because to much prob. mass is moved to all the zeros
  - Add-one is a poor method for smoothing

Estimators: Lidstone's Law

- Instead of adding 1, add some (normally smaller) positive value λ

$$P_{Lid}(w_1 \ldots w_n) = (C(w_1 \ldots w_n) + \lambda) \; / \; (N + B\lambda)$$

  - Most widely use λ is ½
  - In practice
    - Avoid giving too much probability space to unseen events by choosing a small λ
    - Problem
      - What is a good way to guess an appropriate value for λ
      - Discounting using Lidstone 's law always gives probability estimates linear in the MLE frequency
        - Not a good match to the empirical distribution at low frequencies

Discounting

- Discounting
  - Discount some non-zero counts to get the probability mass that will be assigned to the zero counts
- Many works define smoothing algorithm in terms of discount, d_c

$$d_c = c^*/c \qquad ; c^* = \text{discounted counts, } c = \text{original counts}$$

Witten-Bell discounting

- Slightly more complex than Add-One
- Idea:
  - Think of zero-frequency N-gram as one that just hasn't happened yet. When it does happen, it will be the first time seeing this new N-gram.
- So, prob. of zero-frequency N-gram ≈ prob. of seeing N-gram for the first time.

Key concept: Things Seen Once
Use the count of things you've seen once to help estimate the count of things you've never seen.

- Count of first-time N-grams is just the number of N-gram types we saw in the data
  - Reason: we see each type for the first time exactly once

Witten-Bell discounting (2)

- Total probability mass of all zero N-grams

$$\sum_{i;c_i=0} p_i^* = \frac{T}{N+T}$$

; N = number of tokens
T = number of observed types

  - Why (N+T)??

    → think of as we have one count for each tokens and one count for each new type

  - Note that : for add-one → T = V

    : here → $T \neq V$; T=types we have already seen

    V=total number of possible types we might ever see

- Suppose there are Z n-grams with zero count

$$p_i^* = \frac{T}{(N+T)Z}$$

  - The extra prob. mass comes from discounting the prob. of all the seen N-grams

$$p_i^* = \frac{C_i}{(N+T)}; \quad if \quad C_i > 0$$

Witten-Bell discounting (3)

- Smoothed counts can be represented as

$$C_i^* = \begin{cases} \frac{T}{Z(N+T)} N & ; \text{ if } C_i = 0 \\ \frac{C_i}{N+T} N & ; \text{ if } C_i > 0 \end{cases}$$

New prob. of zero-counted n-gram $p_i^* = \frac{T}{(N+T)Z}$

Discounted prob. of non zero-counted n-gram $p_i^* = \frac{C_i}{(N+T)}; \quad if \quad C_i > 0$

Witten-Bell discounting (4)

- For bigram
  - Prob. of unseen bigram $w_{n-1}w_n \approx$ prob. of seeing a new bigram starting with w_{n-1}

$$\sum_{i;c(w_{i-1}w_i)=0} p_i^*(w_i \mid w_{i-1}) = \frac{T(w_{i-1})}{N(w_{i-1}) + T(w_{i-1})}$$

; N = number of bigram tokens, conditioned on w_{i-1}
T = number of observed bigram types, conditioned on w_{i-1}

- Let Z = total number of bigrams with a given first word that have count zero

$$p_i^*(w_i \mid w_{i-1}) = \frac{T(w_{i-1})}{Z(w_{i-1})(N(w_{i-1}) + T(w_{i-1}))} \; ; \text{ if } C(w_{i-1}w_i) = 0$$

Z(w) = V – T(w)

→ there are total V words, so there are V possible bigrams that begin with w. So, the number of unseen bigram types with a given prefix is (V – number of observed types).

  - For non-zero bigrams

$$\sum_{i;c(w_{i-1}w_i)\neq 0} p_i^*(w_i \mid w_{i-1}) = \frac{C(w_i w_{i-1})}{C(w_{i-1}) + T(w_{i-1})}$$

Witten-Bell discounting (5)

- Reconstructed count matrix

	I	want	to	eat
I	8	1087	0	13
want	3	0	786	0
to	3	0	10	860
eat	0	0	2	0

Original counts

	I	want	to	eat
I	6	740	.68	10
want	2	.42	331	.42
to	3	.69	8	594
eat	.37	.37	1	.37

Add-One Reconstructed count

	I	want	to	eat
I	8	1060	.062	13
want	3	.046	740	.046
to	3	.085	10	827
eat	.075	.075	2	.075

Witten-Bell discounted count

Discounted values are more reasonable than those from add-one smoothing

Good-Turing Discounting

- More complex than Witten-Bell
- Re-estimate the amount of prob. mass to assign to N-grams with zero or low counts by looking at the number of N-grams with higher counts
  - Examine number of N-grams that occur C times (N_c)
    - Refer to this number as freq. of freq. c
  - N_0 → number of n-gram of count 0
  - N_1 → number of n-gram with count 1

- Good-Turing

  $C^* = (C+1)(N_{C+1}/N_C)$ ← (discounting the count value C to C*)

- In practice
  - C* is not used for all C
  - Large counts C are assumed to be reliable

    $C^* = C$ for $C < k$

BackOff

- Introduce in 1987 by Katz
- Discounting can help solving the problem of zero-freq. N-grams
- Another strategy to solve this problem
  - Rely on N-gram hierarchy
  - With no example of a particular N-gram, we can estimate its probability by using (N-1)-gram probability
  - Build N-gram model based on (N-1)-gram model
- In back-off, if we have N-gram counts, we rely solely on that counts and don't interpolate the (N-1)-gram counts at all
- Example of tri-gram backoff

$$\hat{p}(w_i \mid w_{i-2}w_{i-1}) = \begin{cases} P(w_i \mid w_{i-2}w_{i-1}) & ;\text{if } c(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha_1 P(w_i \mid w_{i-1}) & ;\text{if } c(w_{i-2}w_{i-1}w_i) = 0 \\ & \quad \text{and } c(w_{i-1}w_i) > 0 \\ \alpha_2 P(w_i) & ;\text{otherwise} \end{cases}$$

Katz's back-off (2)

- Backing off can work badly in some circumstances
  - If $w_j w_k$ occurs many times, and w_k is a common word (i.e. occurs a lot)
  - But!!! We have never seen trigram $w_i w_j w_k$
  - Ex: ฉันกินน้ำมัน
- By backing-off
  - Instead of routinely back off and estimating $P(w_k|h)$ via bigram estimate $P(w_k|w_j)$
  - We should represent grammatical-zero for $w_i w_j w_k$

Combining Estimators

- Katz's backing-off
  - Different models are consulted in order depending on their specificity
  - Katz's back-off n-gram models
    - Estimation is allowed to back off through progressively shorter histories:

$$P_{bo}(w_i \mid w_{i-n+1} \ldots w_{i-1}) = \begin{cases} d_{w_{i-n+1} \ldots w_i} \dfrac{C(w_{i-n+1} \ldots w_i)}{C(w_{i-n+1} \ldots w_{i-1})} & ;\text{if } C(w_{i-n+1} \ldots w_i) > k \\ \alpha_{w_{i+n+1} \ldots w_{i-1}} P_{bo}(w_i \mid w_{i-n+2} \ldots w_{i-1}) & ;\text{otherwise} \end{cases}$$

    - If the focus n-gram appeared more than *k* times, then n-gram estimate is used.
      - MLE is discounted a certain amount (function of *d*) so that some prob mass is reserved for unseen n-grams whose probability will be estimated by backing off.
      - d: discount, or else there would be no prob. mass to distribute to lower order model

        d is typically the amount of discounting found by Good–Turing estimation => d = C*/C
      - α: normalizing factor: only prob. mass left over in the discounting process is distributed among n-grams that are estimate by backing-off

http://en.wikipedia.org/wiki/Katz's_back-off_model

Interpolation

- Simple linear interpolation (or mixture models)
  - Solving sparseness problem in trigram model by mixing trigram model with bigram and unigram models that suffer less from data sparseness.

$$P_{li}(w_n|w_{n-2}, w_{n-1}) = \lambda_1 P_1(w_n) + \lambda_2 P_2(w_n|w_{n-1}) + \lambda_3 P_3(w_n|w_{n-1},w_{n-2})$$

  where $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$

- General linear interpolation

$$P_{li}(w|h) = \sum_{i=1}^{k} \lambda_i(h) P_i(w|h)$$

  where $\forall h,\ 0 \leq \lambda_i(h) \leq 1$ and $\sum_i \lambda_i(h) = 1$

- λ can be considered as a function of the context

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

  - Training λ can be done by using a version of the EM algorithm.

N-gram model: What to count? [1]

- N-gram
  - Related to word count
  - What should we count??

    In general text …
    - Should we count **punctuation**??
    - Should we treat **capitalized and non-capitalized** as the same word??
    - How should we deal with **inflected form**??

    In spoken language
    - How should we treat **fragments**, i.e. words broken off in the middle??
    - How should we treat **filled pauses**, e.g. “umm”, “uh”, …

N-gram model: What to count? [2]

- Look at the following sentence:
  - Hong Kong's Hang Seng Index tanked nearly 3%, while China's Shanghai Composite Index fell 1.2%.
  - Trump said he would raise tariffs on $250 billion in Chinese exports to 30% from 25% in October.
  - French President Emmanuel Macron announced on Saturday (13th July) that he has approved the creation of a space command.

- For the strings with red-color mark
  - It's not about their literal appearances, it's about their semantic category
    - Person name, number, date, etc.
- Preprocessing of these strings to generalize them to their class could be useful in many tasks

Let's try …

- Coding program for N-gram model
  - Unigram
  - 2-gram
  - 3-gram
  - 4-gram
  - …
- Generating sentences from n-gram model
- Calculating probability of the sentences