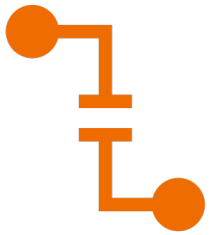
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Tokenization, Language Modeling and Smoothing

Manish Shrivastava

Tokenization



The task of separating ‘Tokens’ in a given input sentence

Example

“This is my brother’s cat”

Should be

“ This is my brother’s cat ”

Or should it be

“ This is my brother ‘ s cat ”

Terms

- Token : a ‘Token’ is a single surface form word
  - Example :
    - My cat is afraid of other cats .
  - Here, ‘cat’ and ‘cats’ are both tokens
    - Giving a total of 8 tokens in the above sentence
    - ‘ . ’ or any punctuation mark is counted as a token

Terms

- Type : Type is a vocabulary word
  - A word which might be present in its root form in the dictionary
  - ‘cat’ is the type for both ‘cat’ and ‘cats’
- The set of all types is the vocabulary of a language
- Ques: What would a high token to type ratio tell you about a language ?

Tokenization

- Identifying individual tokens in a given input
    - Types are not of concern at this point
- Ques: Is the task of tokenization difficult?
    - ??

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Challenges

- Hyphenation
  - I am a hard-working student .
  - We would deal with the state-of-the-art .
  - I am not going to sho-

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Challenges

- Number
  - The value of gravity is 9.8 m/s/s
  - He got 1,000,000 dollars in VC funding .
  - My number is +918451990342
  - Take ½ cups of milk

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Challenges

- Dates
  - My birth date is 03/09/1982
  - I joined on 22nd July

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Challenges

- Abbreviation
  - Dr. S. P. Kishore is the primary faculty of this course
  - We are in IIIT-H campus

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Challenges

- Punctuations
  - This, hands-on experience, is rare in pedagogy .
  - I … uh … not sure how to proceed :-)

Challenges

- URLs
  - The site for course materials for this class is http://tts.iiit.ac.in/~kishore/mediawiki/index.php/NLP:Tokens_and_Words

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Challenges

- Sentencification
  - “This is a presentation. It could also be a video”

Regular Expressions

- Example
  - $ls *.txt
    - All files ending in “.txt”
- Abbreviation
  - “[A-Za-z]+.”
  - “[A-Za-z][A-Za-z]*\.”
    - “ . “ matches any character , hence needs to be escaped to match character “ . “
    - [A-Za-z] matches a single upper- or lower-case character

Regular Expressions

- \s :matches any white-space eg. Tab,newline or space
- \t :tab
- \n :newline
- “^a” :matches strings beginning with letter ‘a’
- [^A-Z] : (NOTE the square braces) any character NOT in the sequence within the braces
- [aeiou] : will match one of the vowels (Think of this as the OR operator within the square braces)
- A||B : matches ‘A’ or ‘B’

Regular Expressions

- Each programming language implements slightly differently
  - Java : Pattern and Matcher classes
  - C : include <regex.h>
  - Perl : inbuilt
  - Python : import re;

N-Grams and Zipf's Law

Basic Idea:

- Examine short sequences of words
- How likely is each sequence?
- “Markov Assumption” – word is affected only by its “prior local context” (last few words)

Example

- The boy ate a chocolate
- The girl bought a chocolate
- The girl then ate a chocolate
- The boy bought a horse

- Can we figure out how likely is the following sentence
    - The boy bought a chocolate

"Shannon Game"

- Claude E. Shannon. "Prediction and Entropy of Printed English", *Bell System Technical Journal* 30:50-64. 1951.
- Predict the next word, given (*n-1)* previous words
- Determine probability of different sequences by examining training corpus

Forming Equivalence Classes (Bins)

- "*n-gram*" = sequence of n words
  - bigram
  - trigram
  - four-gram or quadrigram
- Probabilities of n-grams
  - Unigram

$$p(w) = \frac{c(w)}{N}$$

  - Bigram

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

  - Trigram

$$P(w_i \mid w_{i-1}, w_{i-2}) = \frac{c(w_i, w_{i-1}, w_{i-2})}{c(w_{i-1}, w_{i-2})}$$

Maximum Likelihood Estimation

- The boy bought a chocolate
  - Unigram Probabilities
    - (4/21)*(2/21)*(2/21)*(4/21)*(3/21)
    - (4*2*2*4*3)/21^5 = 0.000047
  - Bi-gram Probabilities
    - <The boy> <boy bought> <bought a> <a chocolate>
    - (2/4)*(2/4)*(2/2)*(3/4) = 0.1875
- Data
  - The boy ate a chocolate
  - The girl bought a chocolate
  - The girl then ate a chocolate
  - The boy bought a horse

Reliability vs. Discrimination

"large green _____"

tree? mountain? frog? car?

"swallowed the large green _____"

pill? candy?

Reliability vs. Discrimination

- larger n: more information about the context of the specific instance (greater discrimination)
- smaller n: more instances in training data, better statistical estimates (more reliability)

Selecting an *n*

Vocabulary (V) = 20,000 words

n	Number of bins
2 (bigrams)	400,000,000
3 (trigrams)	8,000,000,000,000
4 (4-grams)	1.6×10^{17}

Statistical Estimators

- Given the observed training data …
  - How do you develop a model (probability distribution) to predict future events?
  - Language Modeling
    - Predict Likelihood of sequences

Maximum Likelihood Estimation

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

- Estimate sequence probabilities using “counts” or frequencies of sequences
- Problems
  - Sparseness
  - What do you do when unknown words are seen??

Example

- Data
  - The boy ate a chocolate
  - The girl bought a chocolate
  - The girl then ate a chocolate
  - The horse bought a boy
- The boy bought a chocolate
  - Unigram Probabilities
    - (4/21)*(2/21)*(2/21)*(4/21)*(3/21)
    - (4*2*2*4*3)/21^5 = 0.000047
  - Bi-gram Probabilities
    - <The boy> <boy bought> <bought a> <a chocolate>
    - (2/4)*(0/4)*(2/2)*(3/4) = **0**

Approximating Shakespeare

- Generating sentences with random unigrams...
  - Every enter now severally so, let
  - Hill he late speaks; or! a more to leg less first you enter
- With bigrams...
  - 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.
- Trigrams
  - Sweet prince, Falstaff shall die.
  - This shall forbid it should be branded, if renown made it empty.

- Quadrigrams
  - What! I will go seek the traitor Gloucester.
  - Will you not tell me who I am?
  - What's coming out here looks like Shakespeare because it ***is*** Shakespeare
- Note: ***As we increase the value of N, the accuracy of an n-gram model increases, since choice of next word becomes increasingly constrained***

N-Gram Training Sensitivity

- If we repeated the Shakespeare experiment but trained our n-grams on a Wall Street Journal corpus, what would we get?
- Note: ***This question has major implications for corpus selection or design***

WSJ is *not* Shakespeare: Sentences Generated from WSJ

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

Evaluation and Data Sparsity Questions

- Perplexity and entropy: how do you ***estimate*** how well your language model fits a corpus once you're done?
- Smoothing and Backoff : how do you handle unseen n-grams?

Perplexity and Entropy

- Information theoretic metrics
  - Useful in measuring how well a grammar or language model (LM) models a natural language or a corpus
- Entropy: With 2 LMs and a corpus, which LM is the better match for the corpus? How much information is there about what the next word will be? More is better!
  - For a random variable ***X*** ranging over e.g. bigrams and a probability function ***p(x),*** the entropy of X is the expected negative log probability

$$H(X) = -\sum_{x=1}^{x=n} p(x)\log_2 p(x)$$

- Entropy is the lower bound on the # of bits it takes to encode information e.g. about bigram likelihood

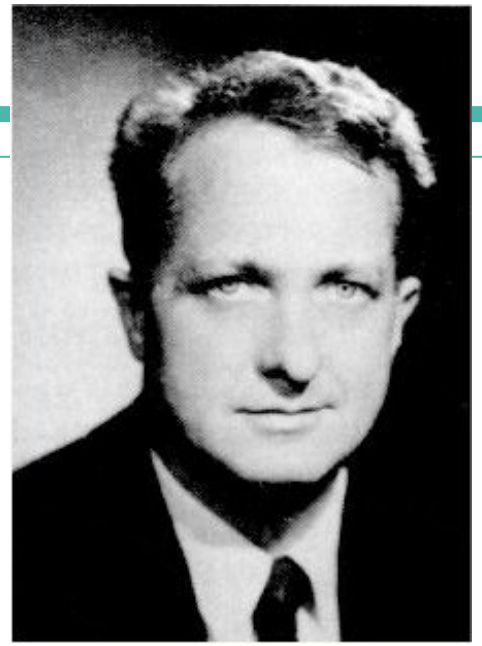
- Perplexity
  - At each choice point in a grammar
    - What are the average number of choices that can be made, weighted by their probabilities of occurrence?
    - I.e., Weighted average branching factor
  - How much probability does a grammar or language model (LM) assign to the sentences of a corpus, compared to another LM? The more information, the lower perplexity

$$PP(W) = 2^{H(W)}$$

Some Useful Observations

- There are 884,647 tokens, with 29,066 word form types, in an approximately one million word Shakespeare corpus
    - Shakespeare produced 300,000 bigram types out of 844 million possible bigrams: so, ***99.96% of the possible bigrams were never seen (have zero entries in the table)***
- 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
- Some zeroes in the table are really zeros But others are simply low frequency events you haven't seen yet. How to address?

Zipf's Law



George Kingsley Zipf
1902-1950

- Frequency of occurrence of words is inversely proportional to the rank in this frequency of occurrence.
- When both are plotted on a log scale, the graph is a straight line.

Zipf Distribution

- The Important Points:
  - a few elements occur *very frequently*
  - a medium number of elements have medium frequency
  - many elements occur *very infrequently*

Zipf Distribution

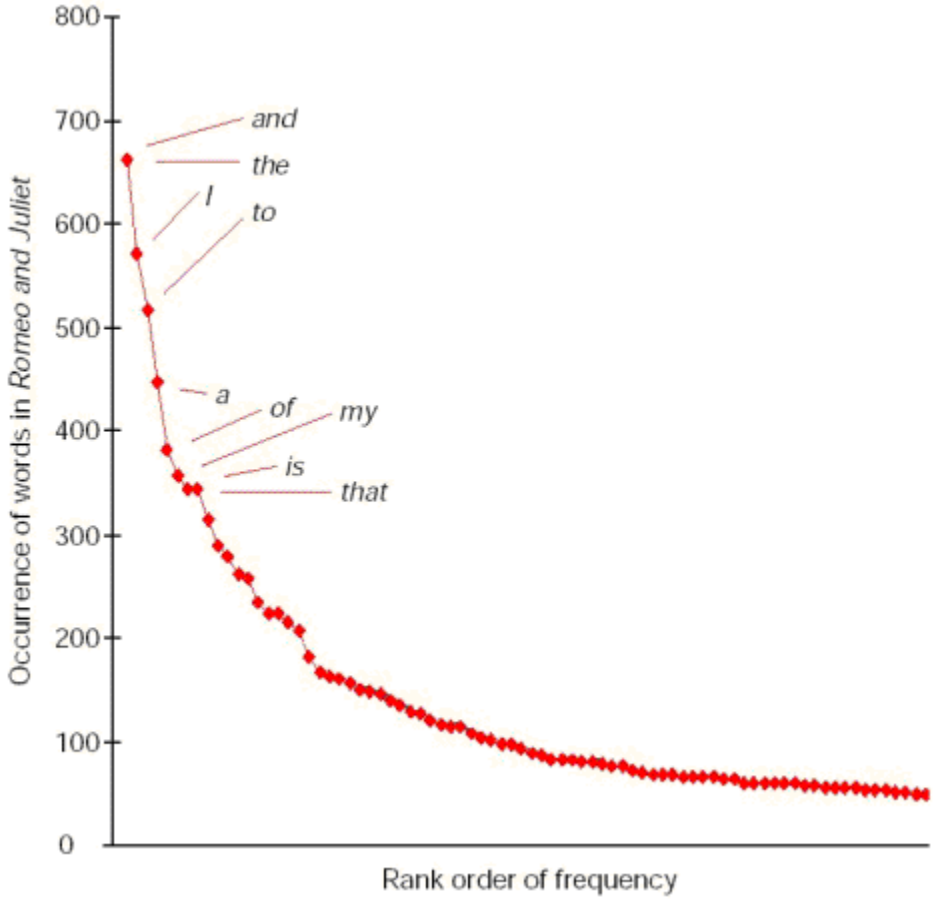
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

The product of the frequency of words (f) and their rank (r) is approximately constant

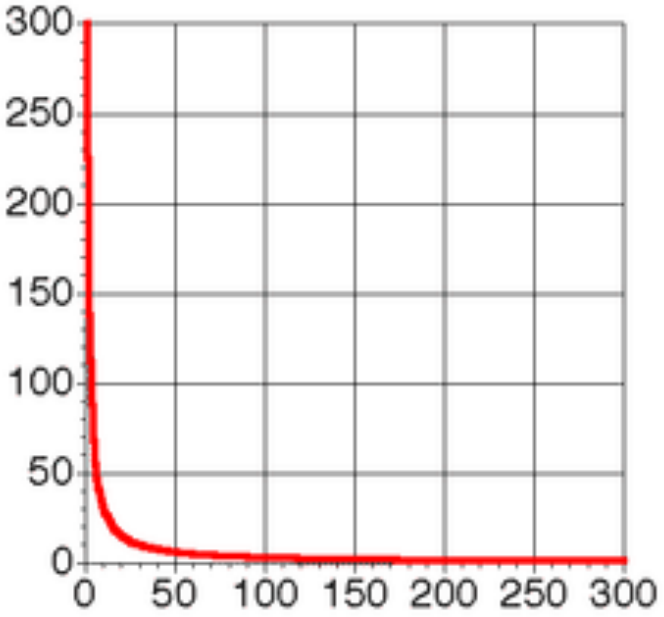
Rank = order of words' frequency of occurrence

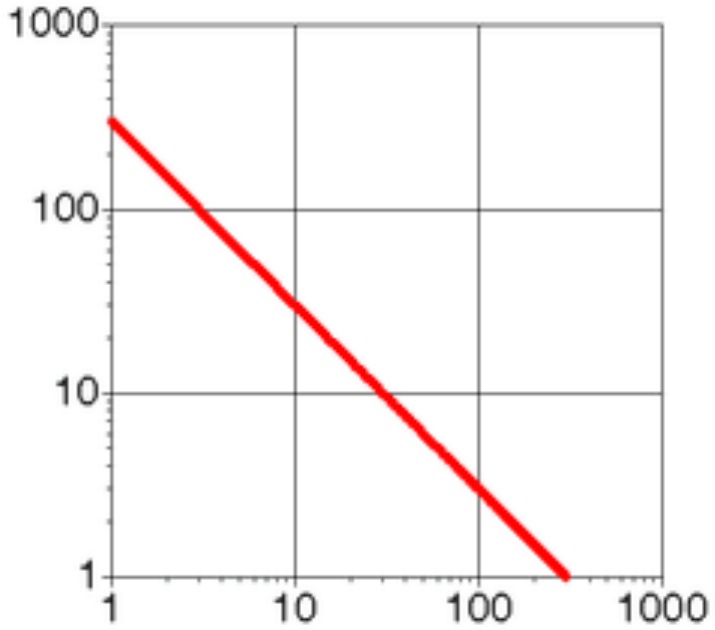
$$f = C * 1/r$$

$$C \cong N/10$$



Zipf Distribution
(Same curve on linear and log scale)





What Kinds of Data Exhibit a Zipf Distribution?

- Words in a text collection
  - Virtually any language usage
- Library book checkout patterns
- Incoming Web Page Requests (Nielsen)
- Outgoing Web Page Requests (Cunha & Crovella)
- Document Size on Web (Cunha & Crovella)

Smoothing

- Laplace smoothing (add-one)
- Good Turing Estimation
- Back off and Interpolation
    - Linear Interpolation
    - Katz Backoff
- Advanced Smoothing Methods
    - Kneser-Ney Smoothing
    - Witten-Bell Smoothing

Laplace Smoothing

- The oldest smoothing method available
- Each unseen n-gram is given a very low estimate
- **Probability 1/(N+B) or Frequency (estimated) N/(N+B)**
- **N is the number of seen n-grams and B is the number of possible n-grams**

Interpolation

- Discounting algorithms can help solve the problem of zero frequency N-grams.
- Additional knowledge that is not used:
  - If trying to compute $P(w_n|w_{n-1}w_{n-2})$ but we have no examples of a particular trigram $w_{n-2}w_{n-1}w_n$
  - Estimate trigram probability based on the bigram probability $P(w_n|w_{n-1})$.
  - If there are no counts for computation of bigram probability $P(w_n|w_{n-1})$, use unigram probability $P(w_n)$.

- There are two ways to rely on this N-gram “hierarchy”:
  - Backoff, and
  - Interpolation

Backoff vs. Interpolation

- Backoff:
  - Relies solely on the trigram counts. When there is a zero count evidence of a trigram then the backoff to lower N-gram.
- Interpolation:
  - Probability estimates are always mixed from all N-gram estimators:
    - Weighted interpolation of trigram, bigram and unigram counts.
    - Simple Interpolation – Linear Interpolation

Linear Interpolation

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

$$\sum_i \lambda_i = 1$$

- Slightly more sophisticated version of linear interpolation with context dependent weights.

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

$$\sum_i \lambda_i(w_{n-2}^{n-1}) = 1$$

Computing Interpolation Weights λ

- Weights are set from held-out corpus.
    - Held-out corpus is additional training corpus that is NOT used to set the N-gram counts but to set other parameters like in this case.
- Choosing λ values that maximize the estimated interpolated probability for example with EM algorithm (iterative algorithm discussed in latter chapters).

Backoff

- Interpolation is simple to understand and implement
- There are better algorithms like backoff N-gram modeling.
  - Uses Good-Turing discounting based on Katz and also known as Katz backoff.

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

- Equation above describes a recursive procedure. Computation of P*, the normalizing factor a, and other details are discussed in next section.

Trigram Discounting with Interpolation

- The w_i, w_{i-1}, w_{i-2} for clarity are referred as a sequence x, y, z.
- Katz method incorporates discounting as integral part of the algorithm.

$$P_{katz}(z \mid x, y) = \begin{cases} P^*(z \mid x, y) & \text{if } C(x, y, z) > 0 \\ \alpha(x, y) P_{katz}(z \mid y) & \text{else if } C(x, y) > 0 \\ P^*(z) & \text{otherwise} \end{cases}$$

$$P_{katz}(z \mid y) = \begin{cases} P^*(z \mid y) & \text{if } C(y, z) > 0 \\ \alpha(y) P_{katz}(z) & \text{otherwise} \end{cases}$$

Katz Backoff

- Good-Turing method assigned probability of unseen events based on the assumption that they are all equally probable.
- Katz backoff gives us a better way to distribute the probability mass among unseen trigram events, by relying on information from unigrams and bigrams.
    - We use discounting to tell us how much total probability mass to set aside for all the events we haven't seen, and backoff to tell us how to distribute this probability.
    - Discount probability $P^*(.)$ is needed rather than MLE $P(.)$ in order to account for the missing probability mass.
    - α weights are necessary to ensure that when backoff occurs the resulting probabilities are true probabilities that sum to 1.

Discounted Probability Computation

- P^* is defined as discounted (c^*) estimate of the conditional probability of an N-gram.

$$P^*\left(w_n \mid w_{n-N+1}^{n-1}\right) = \frac{c^*\left(w_{n-N+1}^{n}\right)}{c\left(w_{n-N+1}^{n-1}\right)}$$

- Because on average the discounted c* will be less than c, this probability P* will be slightly less than the MLE estimate.

Discounted Probability Computation

- The previous slide computation will leave some probability mass for the lower order N-grams, which is then distributed by the α weights (descirbed in next section).
- The table in the next slide shows the result of Katz backoff bigram probabilities for previous 8 sample words computed from BeRP corpus using the SRILM toolkit.

Advanced Details of Computing Katz backoff α and P^*

- Remaining details of computation of α and P^* are presented in this section.
- β – total amount of left-over probability mass function of the (N-1)-gram context.
    - *For a given (N-1) gram context, the total left-over probability mass can be computed by subtracting from 1 the total discounted probability mass for all N-grams starting with that context.*

$$\beta\left(w_{n-N+1}^{n-1}\right) = 1 - \sum_{w_n : c\left(w_{n-N+1}^{n-1}\right) > 0} P^*\left(w_n \mid w_{n-N+1}^{n-1}\right)$$

    - *This gives us the total probability mass that we are ready to distribute to all (N-1)-gram (e.g., bigrams if our original model was trigram)*

Advanced Details of Computing Katz backoff α and P^* (cont.)

- Each individual (*N*-1)-gram (bigram) will only get a fraction of this mass, so we need to normalize β by the total probability of all the (*N*-1)-grams (bigrams) that begin some *N*-gram (trigram) that has zero count. The final equation for computing how much probability mass to distribute from an *N*-gram to an (*N*-1)-gram is represented by the function α:

$$\alpha\left(w_{n-N+1}^{n-1}\right) = \frac{\beta\left(w_{n-N+1}^{n-1}\right)}{\sum_{w_n : c\left(w_{n-N+1}^{n-1}\right) > 0} P_{katz}\left(w_n \mid w_{n-N+2}^{n-1}\right)}$$

$$= \frac{1 - \sum_{w_n : c\left(w_{n-N+1}^{n-1}\right) > 0} P^*\left(w_n \mid w_{n-N+1}^{n-1}\right)}{1 - \sum_{w_n : c\left(w_{n-N+1}^{n-1}\right) > 0} P^*\left(w_n \mid w_{n-N+2}^{n-1}\right)}$$

Advanced Details of Computing Katz backoff α and P^* (cont.)

- Note that a is a function of the preceding word string, that is, of w_{n-N+1}^{n-1} ; thus the amount by which we discount each trigram (*d*), and the mass that gets reassigned to lower order *N*-grams (α) are recomputed for every (*N*-1)-gram that occurs in any *N*-gram.
- We only need to specify what to do when the counts of an (*N*-1)-gram context are 0, (i.e., when $c\left(w_{n-N+1}^{n-1}\right)=0$) and our definition is complete:

$$P_{katz}\left(w_n \mid w_{n-N+1}^{n-1}\right)= P_{katz}\left(w_n \mid w_{n-N+2}^{n-1}\right) \qquad \text{if } c\left(w_{n-N+1}^{n-1}\right)=0$$

$$P^*\left(w_n \mid w_{n-N+1}^{n-1}\right)=0 \qquad \text{if } c\left(w_{n-N+1}^{n-1}\right)=0$$

$$\beta\left(w_n \mid w_{n-N+1}^{n-1}\right)=1 \qquad \text{if } c\left(w_{n-N+1}^{n-1}\right)=0$$

Advanced Smoothing Methods: Kneser-Ney Smoothing

- Brief introduction to the most commonly used modern *N*-gram smoothing method, the **Interpolated Kneser-Ney** algorithm:
  - Algorithm is based on **absolute discounting** method.
  - It is a more elaborate method of computing revised count c^* than the Good-Turing discount formula.
- Re-visiting Good-Turing estimates of the bigram extended from slide Bi-gram Examples.

c(MLE)	0	1	2	3	4	5	6	7	8	9
c*(GT)	0.0000270	0.446	1.26	2.24	3.24	4.22	5.19	6.21	7.24	8.25
Δ=c-c*	-0.0000270	0.554	0.74	0.76	0.76	0.78	0.81	0.79	0.76	0.75

Advanced Smoothing Methods: Kneser-Ney Smoothing

- Re-estimated counts c^* for greater than 1 counts could be estimated pretty well by just subtracting 0.75 from the MLE count c.
- Absolute discounting method formalizes this intuition by subtracting a fixed (absolute) discount d from each count.
    - The rational is that we have good estimates already for the high counts, and a small discount d won't affect them much.
    - The affected are only the smaller counts for which we do not necessarily trust the estimate anyhow.
- The equation for absolute discounting applied to bigrams (assuming a proper coefficient α on the backoff to make everything sum to one) is:

$$P_{absolute}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{c(w_{i-1}w_i) - D}{c(w_{i-1})}, & \text{if } c(w_{i-1}w_i) > 0 \\ \alpha(w_i) P_{absolute}(w_i) & \text{otherwize} \end{cases}$$

Advanced Smoothing Methods: Kneser-Ney Smoothing

- In practice distinct discount values d for the 0 and 1 counts are computed.
- Kneser-Ney discounting augments absolute discounting with a more sophisticated way to handle the backoff distribution. Consider the job of predicting the next word in the sentence, assuming we are backing off to a unigram model:
  - I can't see without my reading XXXXXX.
- The word "*glasses*" seem much more likely to follow than the word "*Francisco*".
  - But "*Francisco*" is in fact more common, and thus a unigram model will prefer it to "*glasses*".

1. Thus we would like to capture that although "*Francisco*" is frequent, it is only frequent after the word "*San*".
2. The word "*glasses*" has a much wider distribution.

Advanced Smoothing Methods: Kneser-Ney Smoothing

- Thus the idea is instead of backing off to the unigram MLE count (the number of times the word w has been seen), we want to use a completely different backoff distribution!
  - We want a heuristic that more accurately estimates the number of times we might expect to see word w in a new unseen context.
  - The Kneser-Ney intuition is to base our estimate on the number of ***different contexts word w has appeared in***.
  - Words that have appeared in more contexts are more likely to appear in some new context as well.
  - New backoff probability can be expressed as the "***continuation probability***" presented in following expression:

Advanced Smoothing Methods: Kneser-Ney Smoothing

- Continuation Probability:

$$P_{continuation}(w_i) = \frac{\left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}{\sum_{w_i}\left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}$$

- Kneser-Ney backoff is formalized as follows assuming proper coefficient α on the backoff to make everything sum to one:

$$P_{KN}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{c(w_{i-1}w_i) - D}{c(w_{i-1})}, & \text{if } c(w_{i-1}w_i) > 0 \\ \alpha(w_i)\dfrac{\left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}{\sum_{w_i}\left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|} & \text{otherwize} \end{cases}$$

Interpolated vs Backoff form of Kneser-Ney

- Kneser-Ney **backoff** algorithm was shown to be less superior to its **interpolated** version.
  Interpolated Kneser-Ney discounting can be computed with an equation like the following (omitting the computation of β):

$$P_{KN}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i) - D}{c(w_{i-1})} + \beta(w_i)\frac{\left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}{\sum \left|\{w_{i-1} : c(w_{i-1}w_i) > 0\}\right|}$$

- Practical note – it turns out that any interpolation model cab be represented as a backoff model, and hence stored in ARPA backoff format. The interpolation is done when the model is built, thus the 'bigram' probability stored in the backoff format is really 'bigram already interpolated with unigram'.

Witten-Bell Discounting

- Probability mass is shifted around, depending on the context of words
- If $P(w_i \mid w_{i-1},\ldots,w_{i-m}) = 0$, then the smoothed probability $P_{WB}(w_i \mid w_{i-1},\ldots,w_{i-m})$ is higher if the sequence $w_{i-1},\ldots,w_{i-m}$ occurs with many different words w_i

Witten-Bell Smoothing

- Let's consider bi-grams
  - $T(w_{i-1})$ is the number of different words (types) that occur to the right of w_{i-1}
  - $N(w_{i-1})$ is the number of all word occurrences (tokens) to the right of w_{i-1}
  - $Z(w_{i-1})$ is the number of bigrams in the current data set starting with w_{i-1} that do not occur in the training data

Witten-Bell Smoothing

- If $c(w_{i-1},w_i) = 0$

- If $c(w_{i-1},w_i) > 0$

$$P^{WB}(w_i \mid w_{i-1}) = \frac{T(w_{i-1})}{Z(w_{i-1})(N + T(w_{i-1}))}$$

$$P^{WB}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{N(w_{i-1}) + T(w_{i-1})}$$

Witten-Bell Smoothing

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

c_i

	I	want	to	eat	Chinese	food	lunch
I	6	740	.68	10	.68	.68	.68
want	2	.42	331	.42	3	4	3
to	3	.69	8	594	3	.69	9
eat	.37	.37	1	.37	7.4	1	20
Chinese	.36	.12	.12	.12	.12	15	.24
food	10	.48	9	.48	.48	.48	.48
lunch	1.1	.22	.22	.22	.22	.44	.22

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

	I	want	to	eat	Chinese	food	lunch
I	8	1060	.062	13	.062	.062	.062
want	3	.046	740	.046	6	8	6
to	3	.085	10	827	3	.085	12
eat	.075	.075	2	.075	17	2	46
Chinese	2	.012	.012	.012	.012	109	1
food	18	.059	16	.059	.059	.059	.059
lunch	4	.026	.026	.026	.026	1	.026

$$c_i' = \begin{cases} T/Z \cdot \frac{N}{N+T} & \text{if } c_i = 0 \\ c_i \cdot \frac{N}{N+T} & \text{otherwise} \end{cases}$$

Witten-Bell Smoothing

- Witten-Bell Smoothing is more conservative when subtracting probability mass
- Gives rather good estimates
- Problem: If w_{i-1} and w_i did not occur in the training data the smoothed probability is still zero