Corpus Linguistics
N-Gram Models

Symbolic Approach (vs. Statistical Approach)

- Do not explicitly use probabilities in the representation of learned knowledge
  - Systems use / acquire knowledge in the form of rules and make decisions based on this stored information
- The learned knowledge is interpretable by humans as contrasted to matrices of probabilities
- Can account for the higher-level, less rule-governed levels of language (longer range dependencies)

Symbolic Approach to Proper Noun (PN) Categorization

- Goal is to assign each part-of-speech tagged Proper Noun phrase (tagged here as PN) to one of n possible categories
  - the|DT extremist|JJ Harkatul_Jihad|PN group|NN ,|, reportedly|RB backed|VBD by|IN Saudi|NP dissident|NN Osama_bin_Laden|PN
  - e.g. *Person, Company, Location, Organization*, etc.
- Techniques
  - Utilize both lexical knowledge bases & context rules
  - Iterative rule-checking & database look-up
  - No probabilities required, but can be used

Symbolic Approach to PN Categorization (cont'd)

- When a Proper Noun (PN) is encountered:
  1. Check knowledge base -> if there, assign PN cat.
  2. Check prefix & suffix rules -> assign PN cat.
     - Common salutations, titles, corporate abbreviations
  3. Check local context -> assign PN cat.
     - Appositional clues ("*X, former senator*")
     - Relative clause clues ("*Y, who….*")
     - Knowledge Base look-up ("*Z, New York*")
  4. Semantic role in relation to verbs of specific semantic classes
- Order of rules matters

Examples of PN Categorization Rule-Writing

PN	PN Cat.	Clues	Resources
Pat Fahey	*Person*	*‘says’*	*lexicon of verbs*
Pacific National Bank	*Org*	*‘Bank’*	*list of suffixes*

Symbolic NLP + Machine Learning Applications

- Transformation-based **POS tagger** learning
- Decision Tree induction-based **parser**
- Exemplar method for **sense disambiguation**
- Decision Tree induction for **anaphora resolution**
- Explanation-based **discourse schema** learning
- Case-based extraction of **pragmatic knowledge**

Statistical Methods (vs. Symbolic Approach)

- Relevant knowledge can be acquired directly from data
  - Use n-grams, Hidden Markov Models, Finite State Automata, or Statistical Parsing Model
  - Attach probabilities to observed “events” in the data, as examples of the feature being learned
- Probabilities are based on training data from corpus
  - If the corpus is pre-annotated with the feature being learned, learning consists of counting observed instances in the data
  - If the corpus is not annotated, estimation strategies are used; new test samples analyzed; correction done; and new annotation used on further samples
- Probability estimates provide a continuous ranking of alternative analyses (for use by next level of processing)

Applications of Statistical NLP

- N-gram model for **speech** recognition
- Probabilistic **Part-of-Speech** Tagging
- Probabilistic Context Free Grammar for **parsing**
- Supervised learning using a Bayesian Probabilistic algorithm for **sense disambiguation**
- Unsupervised learning using MRDs for **sense disambiguation**

What is Corpus Linguistics?

- A methodology to process text ….. AND…..
- Resource provision of (possibly annotated) text

- The Corpus is a collection of text
  - Utilizes a maximally representative sample of machine-readable text of a language or a particular variety of text or language
- Text is usually annotated, or the goal of the research is to annotate it
- Can search, retrieve, calculate, and sort the data in the corpus
- Needed for both symbolic and statistical approaches

Questions to keep in mind for any NLP application:

- Will you need a corpus?
- What are the requisite characteristics of the corpus?
- What type of annotations are needed in this corpus?
- Is there an appropriate annotated corpus already available for your use?

Questions … : (cont’d)

- Will a human or a computer do the annotations?
- What tag-set will you use?
- Does the tag-set need to be specialized for your application or domain?
- If computer annotates, how will you train the system?

Preliminary Text Processing Required :

- Find the words:
  - Filter out ‘junk data’
    - Formatting / extraneous material
    - First be sure it doesn’t reveal important information
  - Deal with upper / lower case issues
  - Tokenize
    - Decide how you define a ‘word’
    - How to recognize and deal with punctuation
      - Apostrophes
      - Hyphens
      - Periods

Preliminary Processing Required: (cont'd)

- Word segmentation
  - No white space in Japanese language
  - Compound words – *"Lebensversicherungsgesellschaftsangestellter"*
- Additional issues if OCR'd data or speech transcripts
- Morphology (To stem or not to stem?)
  - Depends on the application

Preliminary Processing Required: (cont'd)

- May also find additional properties of the text:
  - Mark up the text using chosen tag set:
    - Tokens (words, punctuation, etc.)
    - Part of Speech
    - Named entities
    - Phrase / Clause
    - Sentence
    - Semantics
      - Sense Identification
      - Referent Resolution tagging
    - Discourse Components

N-gram Models

- The n-gram model uses the previous N – 1 ‘*things*’ to predict the next one
  - Can be letters, words, parts-of-speech, etc
- Based on context-sensitive likeliness of occurrence
- We use n-gram word prediction more frequently than we are aware
  - Finishing someone else’s sentence for them
  - Deciphering hard-to-read handwriting
  - Correcting spelling mistakes
- *Do all sounds / letters / words have an equal likelihood of being the next sound / letter / word in a particular situation?*
  - No! so incorporate normal frequency of occurrence
    - The relative frequency used to assign a probability distribution across potential next words

N-gram Models

- Look not just at individual relative frequencies, but at **conditional probability of a word given the previous words**
- Not really possible for long sequences, so we simplify and approximate
  - Probability given just the single 1 or 2 previous words
  - Known as the Markov assumption
- **Bigram Model** – approximates the probability of a word given all the previous words by the conditional probability of the preceding word

N-gram Essential Facts

1. N-gram models become increasingly accurate as the value of N is increased
   – Quadrigams are more accurate than Trigrams, which are more accurate than Bigrams, but are seldom used because of the computational cost and the scarcity of examples of the longer length

2. N-gram models are very dependent on their training corpus, in terms of:
   - Genre
   - Size

Training & Testing

- Typical NLP application wants to find occurrences of some property in text
- Selection of corpus with examples of the property is key decision
  - Not too general – won't reflect a domain / genre
  - Not too specific – produces over-trained probabilities
- Assumes the corpus has been manually tagged and can therefore serve as the gold standard for testing
- Divide corpus into
  - **Training set** for determining the statistical parameters
  - Use these parameters to compute probabilities on **test set**

Word Counting in Corpora

- After corpus preparation, additional decisions
  - Ignore capitalization at beginning of sentence? Is “They” the same word as “they”?
  - Ignore other capitalization? is “Company” the same word as “company”
  - Stemming? Is “cat” the same word as “cats”
- Terminology for word occurrences:
  - Tokens – the total number of words
  - Type – the number of distinct words, not counting repetitions
    - The following sentence from the Brown corpus has 16 tokens and 14 types:
      They picnicked by the pool, then lay back on the grass and looked at the stars.

N-Gram probabilities

- For N-Grams, we need the conditional probability:

  P(<next word> | <preceding word sequence of length n>)

  e.g. P (*the* | *They picnicked by*)

- We define this as
  - the observed frequency (count) of the whole sequence divided by
  - the observed frequency of the preceding, or initial, sequence (sometimes called the maximum likelihood estimation (MLE):

    P(<next word> | <preceding word sequence of length n>)

    = Count (<preceding word sequence> <next word>)

    / Count (<preceding word sequence>)

    Count (*They picnicked by the*) / Count (*They picnicked by*)

- Normalization
  - divide the ratio, aka the relative frequency, by the total word counts to get a probability between 0 and 1

Markov Assumption

- Counting all the long sequences in a corpus is too compute intensive, so we make the assumption that we can predict the next word based on a short piece of the past
- Bigrams often used.
  - Instead of computing

    P (*the* | *They picnicked by*)

    we estimate this probability with the bigram

    P (*the* | *by*)

Example of Bigrams

- Example mini-corpus of three sentences, where we have sentence detection and we include the sentence tags in order to represent the beginning and end of the sentence.

  <S> I am Sam </S>

  <S> Sam I am </S>

  <S> I do not like green eggs and ham </S>

- Bigram probabilities:

  P (I | <S>) = 2/3 = .67

  P (</S> | Sam) = ½ = .5

  P (Sam | <S>) = 1/3 = .33

  P (Sam | am) = ½ = .5

  P (am | I) = 2/3 = .67

Using N-Grams for sentences

- For a bigram grammar $\prod_{k=1}^{n} P(w_k \mid w_{k-1})$
  - –P(sentence) can be approximated by multiplying all the bigram probabilities in the sequence
- Example:

P(I want to eat Chinese food) =

P(I | <S>) P(want | I) P(to | want) P(eat | to)

P(Chinese | eat) P(food | Chinese)

Bigrams from the restaurant corpus

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

Examples due to Rada Mihalcea

Additional Bigrams

<S> I	.25	Want some	.04
<S> I'd	.06	Want Thai	.01
<S> Tell	.04	To eat	.26
<S> I'm	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don't	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

Computing Sentence Probabilities

- 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) = .25×.32×.65×.26×.001×.60 = .000080
- vs.
- P(I want to eat Chinese food) = .00015
- Probabilities seem to capture “syntactic" facts, “world knowledge"
  - eat is often followed by a NP
  - British food is not too popular
- N-gram models can be trained by counting and normalization

In-Class Exercise

Metrics using Corpus Statistics

- Entropy
  - A measure of information
  - A measure of the degree to which a model captures regularities in the corpus by
    - E.g., how predictive a given N-gram model is about what the next word could be
- Perplexity
  - Generalization of entropy that provides a measure of the inherent complexity of a corpus by successively approximating *n*-gram language models of greater and greater *n*
  - A measure of the size of the set of words from which the next word is chosen given the history of words
  - Perplexity of a Language Model depends on domain & genre

Perplexity of Trigram Models

Domain	Perplexity
Radiology	20
Emergency Medicine	60
Journalism	105
General English	247

http://cslu.cse.ogi.edu/HLTsurvey/ch1node8.html#SECTION163

Smoothing Techniques

- Every N-gram training matrix is sparse, even for very large corpora (see Zipf's law later)
  - There are words that don't occur in the training corpus that may occur in future text
  - These are known as the unseen words

- Solution: estimate the likelihood of unseen N-grams

Smoothing

- Add-one smoothing
  - Given: $P(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$
  - Add 1 to each count: $P(w_n|w_{n-1}) = [C(w_{n-1}w_n) + 1] / [C(w_{n-1}) + V]$
- Backoff Smoothing for higher-order N-grams
  - Notice that:
    - N-grams are more precise than (N-1)grams
    - But also, N-grams are more sparse than (N-1) grams
  - How to combine things?
    - Attempt N-grams and back-off to (N-1) if counts are not available
    - E.g. attempt prediction using 4-grams, and back-off to trigrams (or bigrams, or unigrams) if counts are not available

N-gram Application - Spell Correction

- Frequency of spelling errors in human typed text varies
  - 0.05% of the words in carefully edited journals
  - 38% in difficult applications like telephone directory lookup
- Optical character recognition
  - Higher **error** rates than human typists
  - Make different kinds of errors

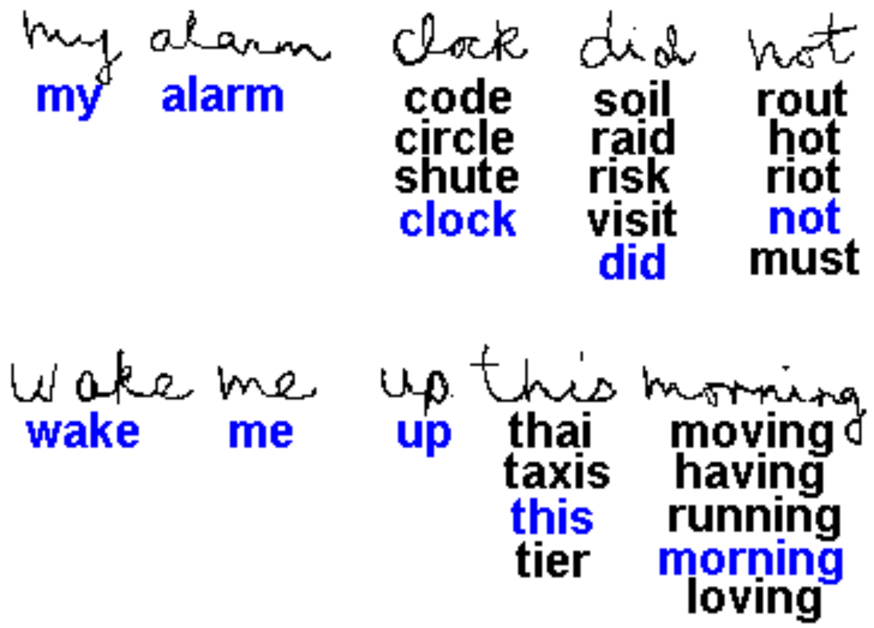
Spelling Error Detection & Correction Types

- **Non-word error detection**
  - Detecting spelling errors that result in non-words
  - *mesage -> message* by looking only at the **word** in isolation
  - Simple lexicon look-up
  - May fail to recognize an error (**real-word errors**)
    - Typographical errors e.g. *there* for *three*
    - Homonym or near-homonym e.g. *dessert* for *desert*, or *piece* for *peace*

- **Context-dependent error detection** and correction
  - Using the context to detect and correct spelling errors
  - Real-word errors are most difficult and would benefit most from n-gram analysis which utilizes context

N-gram Analysis of Handwritten Sentence

- Results of offline isolated word recognition
- Lists of up to top 5 choices of the handwritten word recognizer, with correct choice highlighted
- Using language models with collocational (*alarm clock*) & syntactic (POS) information, correct sentence is extracted:



Language Models

- Language models are a closely related idea that use the N-Gram prediction of the next word to predict sequences of words
- N-Gram language models were first used in large vocabulary speech recognition systems to
  - Provide the recognizer with an a-priori likelihood **P(W)** of a given word sequence **W**.
- The N-Gram language model is usually derived from large training texts that share the same language characteristics as the expected input.
  - This information complements the acoustic model that models the articulatory features of the speakers.
  - Together, these two components allow a system to compute the most likely input sequence
    $\mathbf{W'} = \mathbf{argmax}_W\ \mathbf{P(W|O)}$, where **O** is the input signal observations
    as $\mathbf{W'} = \mathbf{argmax}_W\ \mathbf{P(O|W)\ P(W)}$.

A Statistical Technique: Mutual Information (MI)

- N-Grams predict the next word – Mutual Information computes probability of two words occuring in sequence
- A technique for determining which co-occurrences of words are significant collocations.
  - Based on corpus statistics
  - MI is borrowed from information theory
- Given a pair of words, compares probability that the two occur together as a joint event to the probability they occur individually & that their co-occurrences are simply the result of chance
- The more strongly connected 2 items are, the higher will be their MI value

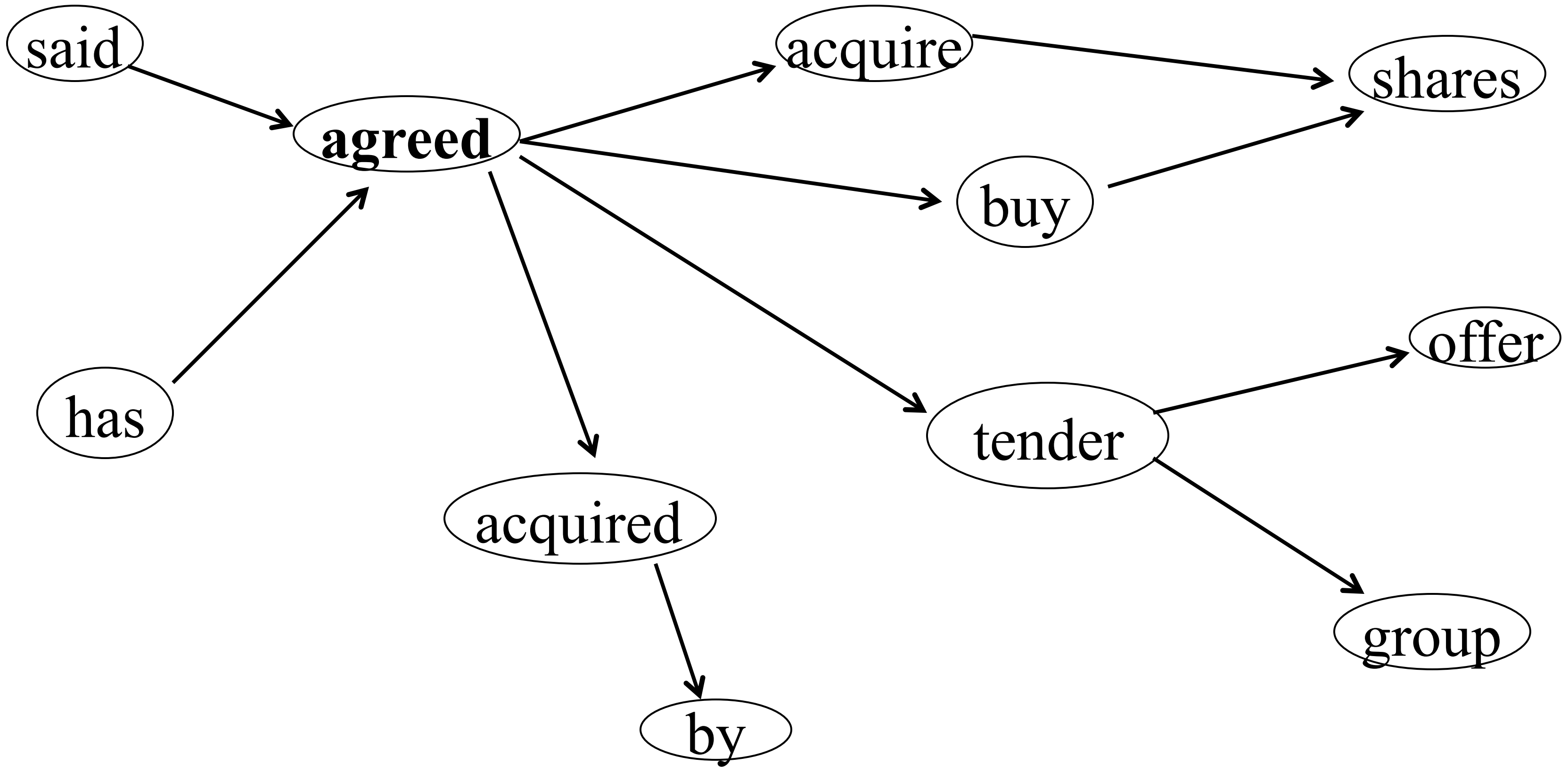
Association Ratio

- Based on work of Church & Hanks (1990), generalizing MI to apply to words in sequence
- Provides robust estimates of word association norms
- P(x) and P(y) are estimated by the number of observations of x and y in a corpus and normalized by N, the size of the corpus
- P(x,y) is estimated by the number of times that x is followed by y in a window of w words, normalized by N
    - Values can be negative or positive

-	0	+
Complementary	Chance	Highly Assoc.

A.R. values based on 145 WSJ articles

x	freq (x)	y	freq (y)	freq (x,y)	AR
Gaza	3	Strip	3	3	14.42
joint	8	venture	4	4	13.00
Chapter	3	11	14	3	12.20
credit	15	card	11	7	11.44
average	22	yield	7	5	11.06
appeals	4	court	47	4	10.45
…..					
said	444	it	346	76	5.02



Highest word associations with *agreed* in WSJ corpus

Uses of Mutual Information or Assocation Ratio

- Lexicographic analysis for dictionary development
- Facilitate development of features to be captured in symbolic applications
  - Idiomatic phrases for MT
  - Semantic word classes for query expansion
  - Lexical candidates for Case Role frames for detecting relations
- Sense disambiguation (both statistical and symbolic approaches)
- Error detection & correction in speech analysis and spell-checking

Zipf's Law

- Rank (r): The numerical position of a word in a list sorted by decreasing frequency (f).
- Zipf (1949) "discovered" that:

$$f \cdot r = k \quad (\text{for constant } k)$$

- If probability of word of rank r is p_r and N is the total number of word occurrences:

$$p_r = \frac{f}{N} = \frac{A}{r} \quad \text{for corpus indp. const. } A \approx 0.1$$

Zipf curve

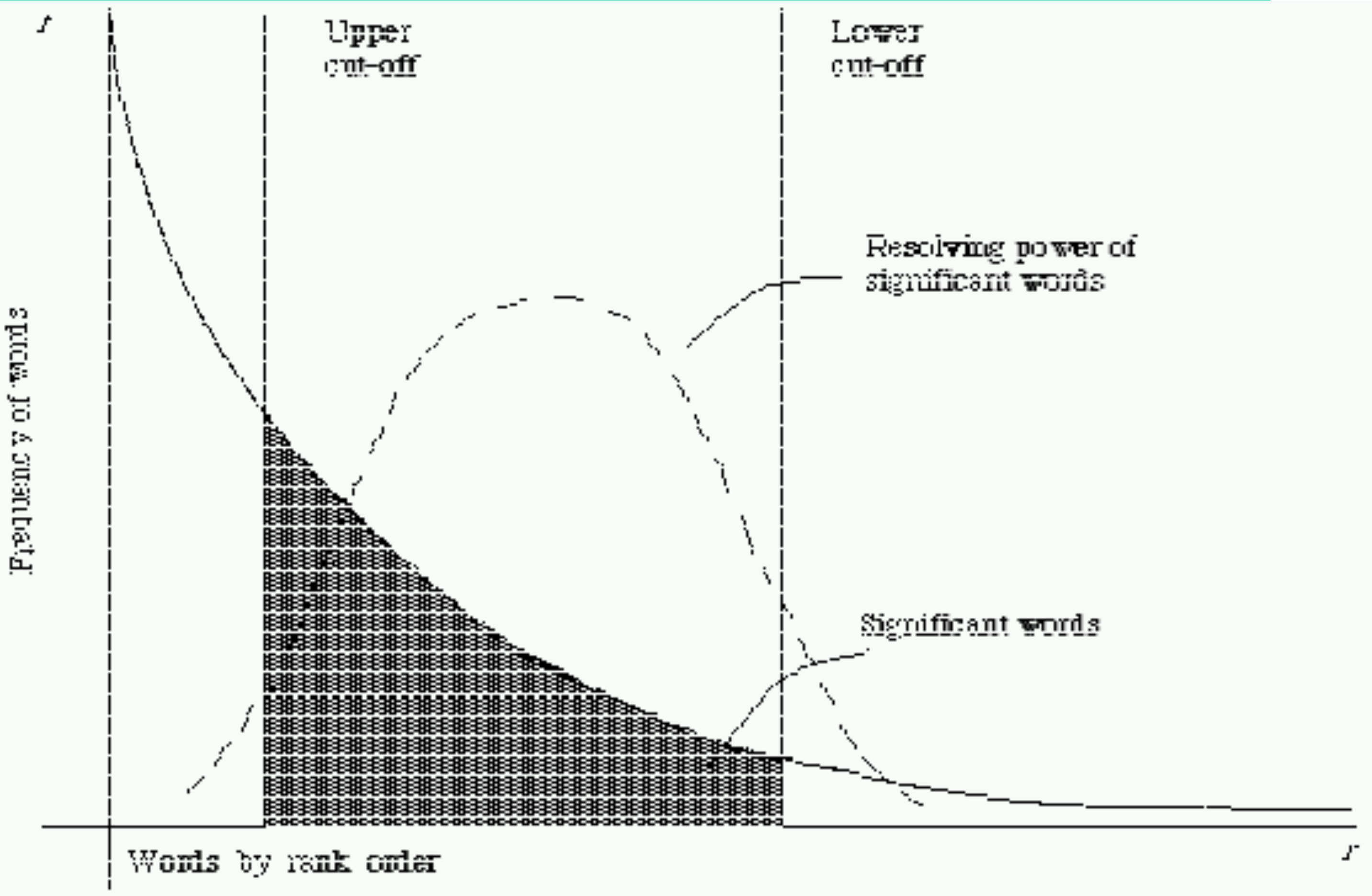


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adaped from Schultz[44] page 120)

Zipf's Law Impact on Language Analysis

- Good News: Stopwords (commonly occurring words such as "the") will account for a large fraction of text so eliminating them greatly reduces size of vocabulary in a text

- Bad News: For most words, gathering sufficient data for meaningful statistical analysis (e.g. for correlation analysis for query expansion) is difficult since they are extremely rare.

Letter-based models – do WE need them? … (fun with human word analysis!)

- Aoccdrnig to rscheearch at an Elingsh uinervtisy, it deosn't mttaer
- in waht oredr the ltteers in a wrod are, olny taht the frist and
- lsat ltteres are at the rghit pcleas. The rset can be a toatl mses
- and you can sitll raed it wouthit a porbelm. Tihs is bcuseae we do
- not raed ervey lteter by ilstef, but the wrod as a wlohe.