Week 2 Lecture
NLP - Text Pre-Processing

CAB431

Professor Yuefeng Li
School of Computer Science
Queensland University of Technology

Learning Objectives

1. Processing Text
   - Text Statistics
   - Tokenizing
   - Stopping and stemming
   - Phrases and N-grams
2. Information Extraction
   - Named Entity
   - HMM - *Hidden Markov Model*
3. Document Structure and Markup
   - HTML tags
   - Hyperlinks

1. Processing Text

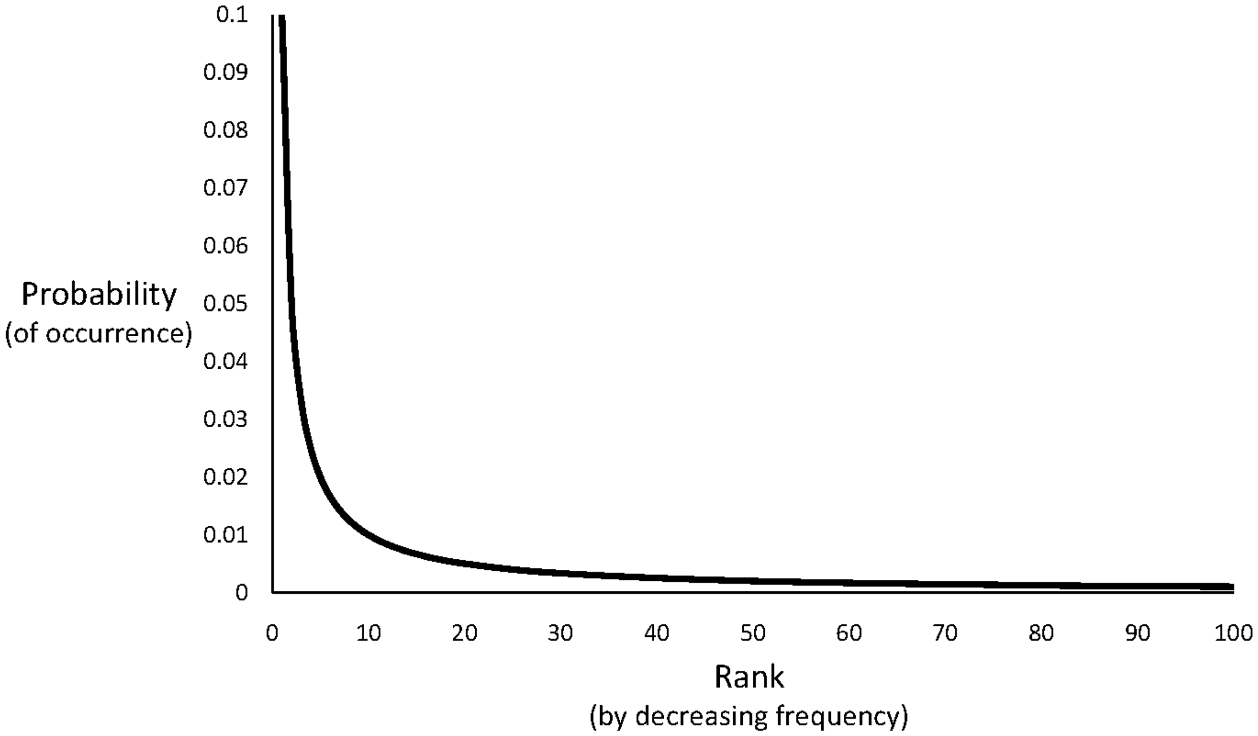
- Converting documents to *index terms*
- Why?
  - Matching the exact string of characters typed by the user is too restrictive
    - i.e., it doesn't work very well in terms of effectiveness
  - Not all words are of equal value in a search
  - Sometimes not clear where words begin and end
    - Not even clear what a word is in some languages
      - e.g., Chinese, Korean

Text Statistics

- Huge variety of words used in text but
- Many statistical characteristics of word occurrences are predictable
  - e.g., distribution of word counts
- Retrieval models and ranking algorithms depend heavily on statistical properties of words
  - e.g., important words occur often in documents but are not high frequency in collection

Zipf's Law

- Distribution of word frequencies is very *skewed*
  - a few words occur very often, many words hardly ever occur
  - e.g., two most common words ("the", "of") make up about 10% of all word occurrences in text documents



Professor Yuefeng Li
School of Computer Science

Vocabulary Growth

- As corpus grows, so does vocabulary size
  - Fewer new words when corpus is already large
- Observed relationship (*Heaps' Law*):

$$v = k \cdot n^{\beta}$$

where v is vocabulary size (number of unique words),

n is the number of words in corpus,

k, β are parameters that vary for each corpus (typical values given are $10 \le k \le 100$ and $\beta \approx 0.5$)

Web Example

- Heaps' Law works with very large corpora
  - new words occurring even after seeing 30 million!
  - parameter values different than typical TREC values
- New words come from a variety of sources
  - spelling errors, invented words (e.g., product, company names), code, other languages, email addresses, etc.
- Web Search must deal with these **large and growing vocabularies**

Document Parsing

- Document parsing involves the recognition of the content and structure of text documents.
- Forming words from sequence of characters is called **tokenizing**.
- Surprisingly complex in English, can be harder in other languages
- **Definition of Words in Early IR systems**:
  - any sequence of alphanumeric characters of length 3 or more
  - terminated by a space or other special character
  - upper-case changed to lower-case

Tokenizing Example

- Example:
  - "Bigcorp's 2007 bi-annual report showed profits rose 10%." ⇒ "**bigcorp 2007 annual report showed profits rose**"
- Too simple for search applications or even large-scale experiments
- Why? Too much information lost
  - **Small decisions in tokenizing can have major impact on effectiveness of some queries**

Tokenizing Problems

- Small words can be important in some queries, usually in combinations
    - xp, ma, pm, ben e king, el paso, master p, gm, j lo, world war II, VW (Volkswagen)
- Both hyphenated and non-hyphenated forms of many words are common
  - Sometimes hyphen is not needed
    - e-bay, wal-mart, active-x, cd-rom, t-shirts
  - At other times, hyphens should be considered either as part of the word or a word separator
    - winston-salem, mazda rx-7, e-cards, pre-diabetes, t-mobile, spanish-speaking

Tokenizing Problems cont.

- Special characters are an important part of tags, URLs, code in documents
- Capitalized words can have different meaning from lower case words
  - Bush, Apple
- Apostrophes can be a part of a word, a part of a possessive, or just a mistake
  - rosie o'donnell, can't, don't, 80's, 1890's, men's straw hats, master's degree, england's ten largest cities, shriner’s.

Tokenizing Problems

- Numbers can be important, including decimals
  - nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358
- Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
  - I.B.M., Ph.D., cs.umass.edu, F.E.A.R.

Note: tokenizing steps for queries **must be identical** to steps for documents

Tokenizing Process

- First step is to use parser (for a specific markup language, e.g. HTML) to identify appropriate parts of the document to tokenize
- Defer complex decisions to other components
  - word is any sequence of alphanumeric characters, terminated by a space or special character, with everything converted to lower-case
  - everything indexed
  - example: 92.3 → 92 3 but search finds documents with 92 and 3 adjacent
  - incorporate some rules to reduce dependence on query transformation components
    - Examples of rules used with TREC
      - Apostrophes in words ignored
        - o’connor → oconnor bob’s → bobs
      - Periods in abbreviations ignored
        - I.B.M. → ibm Ph.D. → phd

Example


```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
- <newsitem xml:lang="en" date="1997-08-06" id="root" itemid="783802">
    <title>UK: Britain's Channel 5 to broadcast Fashion Awards.</title>
    <headline>Britain's Channel 5 to broadcast Fashion Awards.</headline>
    <dateline>LONDON 1997-08-06</dateline>
  - <text>
      <p>Channel 5, Britain's newest terrestrial television channel, on Wednesday said it struck a
        three-year deal with the British Fashion Council for exclusive rights to broadcast the
        Lloyds Bank British Fashion Awards.</p>
      <p>"Fashion is now a defining part of British popular culture and this was an unmissable
        opportunity for 5 to give the British Fashion Awards its rightful place and profile on
        television," Adam Perry, Channel 5's controller of events, said in a statement.</p>
      <p>Initial, the independent production company that will produce coverage of the awards,
        said it aims to bring in top celebrities and world-class performers for the event.</p>
      <p>The British Fashion Awards will be held on October 22 at London's Royal Albert Hall.</p>
      <p>-- London Advertising Newsdesk +44 171 542 2815</p>
    </text>
    <copyright>(c) Reuters Limited 1997</copyright>
</newsitem>
```


```
>>> import string
>>> line ="<p>The British Fashion Awards will be held on October 22 at London's Royal Albert Hall.</p>“
>>> line = line.replace("<p>", "").replace("</p>", "")
>>> line = line.translate(str.maketrans(",", string.digits)).translate(str.maketrans(string.punctuation, ' '*len(string.punctuation)))
>>> line
'The British Fashion Awards will be held on October  at London s Royal Albert Hall '
>>> words=line.split()
>>> words
['The', 'British', 'Fashion', 'Awards', 'will', 'be', 'held', 'on', 'October', 'at', 'London', 's', 'Royal', 'Albert', 'Hall']
>>> terms=[word.lower() for word in words if len(word)>2]
>>> terms
['the', 'british', 'fashion', 'october', 'royal', 'hall', 'will', 'albert', 'london', 'held', 'awards‘]
```

Stopping

- Function words (determiners, prepositions) have little meaning on their own; and
- High occurrence frequencies
- Treated as *stopwords* (i.e., removed)
    - reduce index space, improve response time, improve effectiveness
- Can be important in combinations
    - e.g., “to be or not to be”

Stopping cont.

- Stopword list can be created from high-frequency words or based on a standard list.
- Lists are customized for applications, domains, and even parts of documents
  - e.g., “click” is a good stopword for anchor text
- **Best policy** is to index all words in documents, make decisions about which words to use at query time.

Stemming

- Many morphological variations of words
  - *inflectional* (plurals, tenses)
  - *derivational* (making verbs nouns etc.)
- In most cases, these have the same or very similar meanings
- Stemmers attempt to reduce morphological variations of words to a common stem
  - usually involves removing suffixes
- Can be done at indexing time or as part of query processing (like stopwords).

Stemming cont.

- Generally, a small but significant effectiveness improvement
  - can be crucial for some languages
  - e.g., 5-10% improvement for English, up to 50% in Arabic

kita**b**	*a book*
kita**b**i	*my book*
al**kit**a**b**	*the book*
kita**b**uki	*your book* (f)
kita**b**uka	*your book (m)*
kita**b**uhu	*his book*
ka**t**a**b**a	*to write*
ma**kt**a**b**a	*library, bookstore*
ma**kt**a**b**	*office*

Words with the Arabic root **ktb**

Stemming

- Two basic types
  - Dictionary-based: uses lists of related words
  - Algorithmic: uses program to determine related words
- Algorithmic stemmers
  - *suffix-s:* remove ‘s’ endings assuming plural
    - e.g., cats → cat, lakes → lake, wiis → wii
    - Many *false negatives (have different stems)*: triangle → triangular
    - Some *false positives (have the same stem)*: policy → police

Porter Stemmer

- Algorithmic stemmer used in IR experiments since the 70s
- Consists of a series of rules designed to the longest possible suffix at each step
- Effective in TREC
- Produces ***stems* not *words***
- Makes a number of errors and difficult to modify

```
>>> import os
>>> os.chdir ('C:\\2023\\Python_code\\workshop_tutor\\wk_solutions')
>>> os.getcwd()
'C:\\2023\\Python_code\\workshop_tutor\\wk_solutions'
>>> from stemming.porter2 import stem
>>> stems=[stem(term) for term in terms]
>>> stems
['the', 'british', 'fashion', 'octob', 'royal', 'hall', 'will', 'albert', 'london', 'held', 'award']
```

Porter Stemmer

- Example step (1 of 5)

Step 1a:

- Replace ***sses*** by ***ss*** (e.g., stresses → stress).
- Delete ***s*** if the preceding word part contains a vowel not immediately before the ***s*** (e.g., gaps → gap but gas → gas).
- Replace ***ied*** or ***ies*** by ***i*** if preceded by more than one letter, otherwise by ***ie*** (e.g., ties → tie, cries → cri).
- If suffix is ***us*** or ***ss*** do nothing (e.g., stress → stress).

Step 1b:

- Replace ***eed***, ***eedly*** by ***ee*** if it is in the part of the word after the first non-vowel following a vowel (e.g., agreed → agree, feed → feed).
- Delete ***ed***, ***edly***, ***ing***, ***ingly*** if the preceding word part contains a vowel, and then if the word ends in ***at***, ***bl***, or ***iz*** add ***e*** (e.g., fished → fish, pirating → pirate), or if the word ends with a double letter that is not ***ll***, ***ss*** or ***zz***, remove the last letter (e.g., falling→ fall, dripping → drip), or if the word is short, add ***e*** (e.g., hoping → hope).
- Whew!

Porter Stemmer

False positives	*False negatives*
organization/organ	european/europe
generalization/generic	cylinder/cylindrical
numerical/numerous	matrices/matrix
policy/police	urgency/urgent
university/universe	create/creation
addition/additive	analysis/analyses
negligible/negligent	useful/usefully
execute/executive	noise/noisy
past/paste	decompose/decomposition
ignore/ignorant	sparse/sparsity
special/specialized	resolve/resolution
head/heading	triangle/triangular

- Porter2 stemmer addresses some of these issues, e.g., stem('policy') = 'polici', and stem('police') ='polic' by using Porter2.
- Approach has been used with other languages.

Krovetz Stemmer

- Hybrid algorithmic-dictionary
  - Word checked in dictionary
    - If present, either left alone or replaced with “exception”
    - If not present, word is checked for suffixes that could be removed
    - After removal, dictionary is checked again
- Produces words not stems
- Comparable effectiveness
- Lower false positive rate, somewhat higher false negative

Stemmer Comparison

Original text:

Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Porter stemmer:

document describ market strategi carri compani agricultur chemic report predict market share chemic report market statist agrochem pesticid herbicid fungicid insecticid fertil predict sale market share stimul demand price cut volum sale

Krovetz stemmer:

document describe marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic agrochemic pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale

Q: Which stemmer is better?

Phrases and N-grams

- Many queries are 2-3 word phrases.
- Phrases are
  - More precise than single words
    - e.g., documents containing “black sea” vs. two words “black” and “sea”
  - Less ambiguous
    - e.g., “big apple” vs. “apple”
- Can be difficult for ranking when we use them
    - e.g., Given query “fishing supplies”, how do we score documents with
      - exact phrase many times, exact phrase just once, individual words in same sentence, same paragraph, whole document, variations on words?

Phrases

- Text processing issue – how are phrases recognized?
- Three possible approaches:
  - Identify syntactic phrases using a *part-of-speech* (POS) tagger
  - Use word *n-grams*
  - Store word positions in indexes and use *proximity operators* in queries

POS Tagging

- POS taggers use statistical models of text to predict syntactic tags of words
  - Example tags:
    - NN (singular noun), NNS (plural noun), VB (verb), VBD (verb, past tense), VBN (verb, past participle), IN (preposition), JJ (adjective), CC (conjunction, e.g., “and”, “or”), PRP (pronoun), and MD (modal auxiliary, e.g., “can”, “will”).
- Phrases can then be defined as simple noun groups.

Pos Tagging Example

Original text:

Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Brill tagger:

Document/NN will/MD describe/VB marketing/NN strategies/NNS carried/VBD out/IN by/IN U.S./NNP companies/NNS for/IN their/PRP agricultural/JJ chemicals/NNS ,/, report/NN predictions/NNS for/IN market/NN share/NN of/IN such/JJ chemicals/NNS ,/, or/CC report/NN market/NN statistics/NNS for/IN agrochemicals/NNS ,/, pesticide/NN ,/, herbicide/NN ,/, fungicide/NN ,/, insecticide/NN ,/, fertilizer/NN ,/, predicted/VBN sales/NNS ,/, market/NN share/NN ,/, stimulate/VB demand/NN ,/, price/NN cut/NN ,/, volume/NN of/IN sales/NNS ./.

Python nltk

- The POS tagger in the NLTK library outputs specific tags for words. The list of POS tags can be found at
- https://medium.com/@muddaprince456/categorizing-and-pos-tagging-with-nltk-python-28f2bc9312c3

```
> pip3 install nltk
>>> import nltk
>>> from nltk.tokenize import word_tokenize
```

You may need to do

```
>>> nltk.download('punkt')
>>> nltk.download('averaged_perceptron_tagger')
>>> text = word_tokenize("Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.")
>>> text = [x for x in text if len(x)>1]
>>> text
['Document', 'will', 'describe', 'marketing', 'strategies', 'carried', 'out', 'by', 'U.S.', 'companies', 'for', 'their', 'agricultural', 'chemicals', 'report', 'predictions', 'for', 'market', 'share', 'of', 'such', 'chemicals', 'or', 'report', 'market', 'statistics', 'for', 'agrochemicals', 'pesticide', 'herbicide', 'fungicide', 'insecticide', 'fertilizer', 'predicted', 'sales', 'market', 'share', 'stimulate', 'demand', 'price', 'cut', 'volume', 'of', 'sales']
>>> pos_results = nltk.pos_tag(text)
>>> pos_results
[('Document', 'NNP'), ('will', 'MD'), ('describe', 'VB'), ('marketing', 'NN'), ('strategies', 'NNS'), ('carried', 'VBD'), ('out', 'RP'), ('by', 'IN'), ('U.S.', 'NNP'), ('companies', 'NNS'), ('for', 'IN'), ('their', 'PRP$'), ('agricultural', 'JJ'), ('chemicals', 'NNS'), ('report', 'VBP'), ('predictions', 'NNS'), ('for', 'IN'), ('market', 'NN'), ('share', 'NN'), ('of', 'IN'), ('such', 'JJ'), ('chemicals', 'NNS'), ('or', 'CC'), ('report', 'NN'), ('market', 'NN'), ('statistics', 'NNS'), ('for', 'IN'), ('agrochemicals', 'NNS'), ('pesticide', 'JJ'), ('herbicide', 'NN'), ('fungicide', 'JJ'), ('insecticide', 'JJ'), ('fertilizer', 'NN'), ('predicted', 'VBD'), ('sales', 'NNS'), ('market', 'NN'), ('share', 'NN'), ('stimulate', 'JJ'), ('demand', 'NN'), ('price', 'NN'), ('cut', 'NN'), ('volume', 'NN'), ('of', 'IN'), ('sales', 'NNS')]
```

Example Noun Phrases

TREC data		Patent data	
Frequency	*Phrase*	*Frequency*	*Phrase*
65824	united states	975362	present invention
61327	article type	191625	u.s. pat
33864	los angeles	147352	preferred embodiment
18062	hong kong	95097	carbon atoms
17788	north korea	87903	group consisting
17308	new york	81809	room temperature
15513	san diego	78458	seq id
15009	orange county	75850	brief description
12869	prime minister	66407	prior art
12799	first time	59828	perspective view
12067	soviet union	58724	first embodiment
10811	russian federation	56715	reaction mixture
9912	united nations	54619	detailed description
8127	southern california	54117	ethyl acetate
7640	south korea	52195	example 1
7620	end recording	52003	block diagram
7524	european union	46299	second embodiment
7436	south africa	41694	accompanying drawings
7362	san francisco	40554	output signal
7086	news conference	37911	first end
6792	city council	35827	second end
6348	middle east	34881	appended claims
6157	peace process	33947	distal end
5955	human rights	32338	cross-sectional view
5837	white house	30193	outer surface

Word N-Grams

- POS tagging too slow for large collections
- Simpler definition – phrase is any sequence of *n* words – known as *n-grams*
  - *bigram*: 2 word sequence, *trigram*: 3 word sequence, *unigram*: single words
  - N-grams also used at character level for applications such as OCR
- N-grams typically formed from *overlapping* sequences of words
  - i.e. move n-word "window" one word at a time in document

```
stems = ['the', 'british', 'fashion', 'octob', 'royal', 'hall', 'will', 'albert', 'london', 'held', 'award']

>>> bigrams = [stems[i]+' '+stems[i+1] for i in range(len(stems)-1)]
>>> bigrams
['the british', 'british fashion', 'fashion octob', 'octob royal', 'royal hall', 'hall will', 'will albert', 'albert london', 'london held', 'held award']
>>> trigrams = [stems[i]+' '+stems[i+1]+' '+stems[i+2] for i in range(len(stems)-2)]
>>> trigrams
['the british fashion', 'british fashion octob', 'fashion octob royal', 'octob royal hall', 'royal hall will', 'hall will albert', 'will albert london',
'albert london held', 'london held award']
```

N-Grams

- Frequent n-grams are more likely to be meaningful phrases
- N-grams form a Zipf distribution
  - Better fit than words alone
- Could index all n-grams up to specified length
  - Much faster than POS tagging
  - Uses a lot of storage
    - e.g., document containing 1,000 words would contain 3,990 instances of word n-grams of length $2 \leq n \leq 5$

N-gram Examples from Several Disciplines

Field	Unit	Sample sequence	1-gram sequence
Vernacular name			unigram
Order of resulting Markov model			0
Protein sequencing	amino acid	… Cys-Gly-Leu-Ser-Trp …	…, Cys, Gly, Leu, Ser, Trp, …
DNA sequencing	base pair	…AGCTTCGA…	…, A, G, C, T, T, C, G, A, …
Computational linguistics	character	…to_be_or_not_to_be…	…, t, o, _, b, e, _, o, r, _, n, o, t, _, t, o, _, b, e, …
Computational linguistics	word	… to be or not to be …	…, to, be, or, not, to, be, …

2-gram sequence	3-gram sequence
bigram	trigram
1	2
…, Cys-Gly, Gly-Leu, Leu-Ser, Ser-Trp, …	…, Cys-Gly-Leu, Gly-Leu-Ser, Leu-Ser-Trp, …
…, AG, GC, CT, TT, TC, CG, GA, …	…, AGC, GCT, CTT, TTC, TCG, CGA, …
…, to, o_, _b, be, e_, _o, or, r_, _n, no, ot, t_, _t, to, o_, _b, be, …	…, to_, o_b, _be, be_, e_o, _or, or_, r_n, _no, not, ot_, t_t, _to, to_, o_b, _be, …
…, to be, be or, or not, not to, to be, …	…, to be or, be or not, or not to, not to be, …

Professor Yuefeng Li
School of Computer Science

Google N-Grams

- Web search engines index n-grams
- Google sample:

Number of tokens:	1,024,908,267,229
Number of sentences:	95,119,665,584
Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663

- Most frequent trigram in English is "all rights reserved"
- How to select a small set of useful n-grams [2]

2. Information Extraction

- Automatically extract structure from text
  - annotate document using tags to identify extracted structure
- *Named entity recognition*
  - identify words that refer to something of interest in a particular application
  - e.g., people, companies, locations, dates, product names, prices, etc.

Named Entity Recognition

Fred Smith, who lives at 10 Water Street, Springfield, MA, is a long-time collector of **tropical fish.**

<p ><PersonName><GivenName>Fred</GivenName> <Sn>Smith</Sn> </PersonName>, who lives at <address><Street >10 Water Street</Street>, <City>Springfield</City>, <State>MA</State></address>, is a long-time collector of tropical fish.</p>

- Example showing semantic annotation of text using XML tags
- Information extraction also includes document structure and more complex features such as *relationships* and *events*

Approaches for Named Entity Recognition

- ***Rule-based***
  - Uses *lexicons* (lists of words and phrases) that categorize names
    - e.g., locations, peoples' names, organizations, etc.
  - Rules also used to verify or find new entity names
    - e.g., "<number> <word> street" for addresses
    - "<street address>, <city>" or "in <city>" to verify city names
    - "<street address>, <city>, <state>" to find new cities
    - "<title> <name>" to find new names

Approaches for Named Entity Recognition cont.

- Rules either developed manually by trial or using machine learning techniques
- ***Statistical Approach***
  - uses a probabilistic model of the words in and around an entity
  - probabilities estimated using *training data* (manually annotated text)
  - Hidden Markov Model (HMM) is one approach

HMM for Extraction

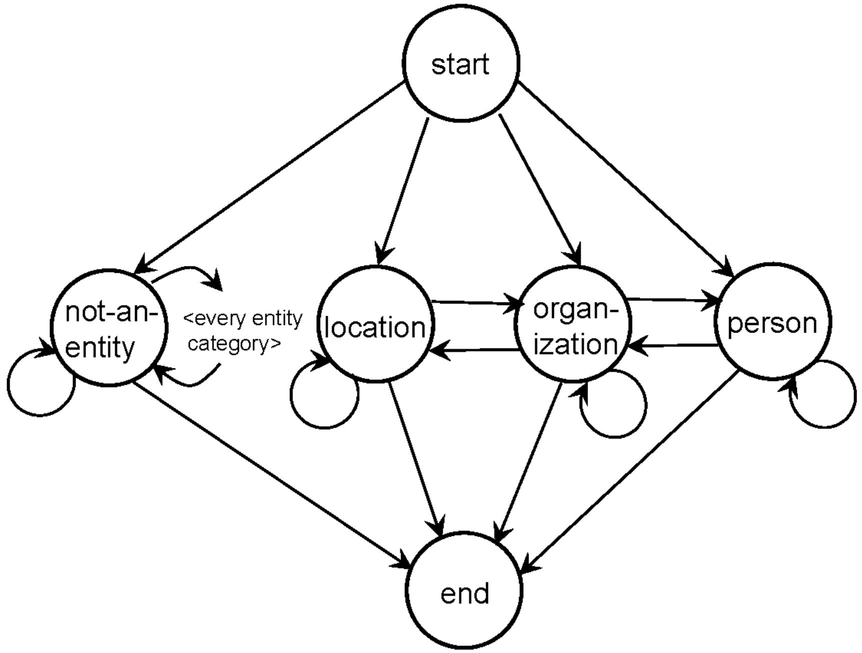
- Resolve ambiguity in a word using *context (the words that surround it)*
  - e.g., “marathon” is a location or a sporting event, “boston marathon” is a specific sporting event
- Model context using a *generative* model of the sequence of words
  - *Markov property*: the next word in a sequence depends only on a small number of the previous words

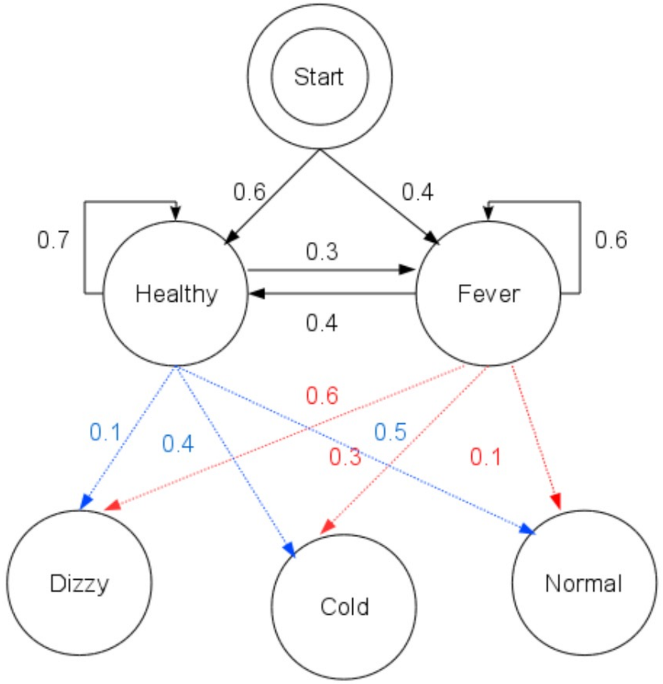
HMM for Extraction cont.

- *Markov Model (MM)* describes a process as a collection of states with transitions between them
  - each transition has a probability associated with it
  - next state depends only on current state and transition probabilities
- *Hidden Markov Model (HMM)*
  - each state has a set of **possible outputs**
  - outputs have probabilities

Professor Yuefeng Li
School of Computer Science

Examples of MM and HMM





- The left is a MM to describe the possible states (categories) of a sentence. It can be extended to an HMM if each state is associated with a probability distribution over words (the output).
- The right one is an HMM which assumes a patient has two states: "Healthy" and "Fever". A doctor cannot see the (hidden) states directly; but the patient can tell the doctor that she/he is "normal", "cold", or "dizzy" (the observations).

HMM for Extraction

- Could generate sentences with this model
- To recognize named entities, find sequence of “labels” that give highest probability for the sentence
  - only the outputs (words) are visible or observed
  - states are “hidden”
  - e.g., <start><name><not-an-entity><location><not-an-entity><end>
- *Viterbi* algorithm used for recognition

Inputs and output of HMM

- Inputs
  - The observation space O, e.g., O is a set of words.
  - The state space S , e.g., S includes entities, such as, “location”, “person”, “organization” and “not-an-entity”; and the initial probabilities p_i of each state s_i or entity.
  - Transition matrix A, where A_{ij} is the transition probability of transiting from state s_i to state s_j.
  - Emission matrix B, where B_{ij} is the probability of observing o_j from state s_i
  - a sequence of observation Y (a sub-set of O in order)
- Output
  - The most likely hidden state sequence X

Inputs and output of HMM cont.

Initial probabilities		p_1	p_2	…	p_K
States		s_1	s_2	…	s_K
	s_1	A_{11}	A_{12}	…	A_{1K}
	s_2	A_{21}	A_{22}	…	A_{2K}
	…	…			…
	s_K	A_{K1}	A_{K2}	…	A_{KK}

$\leftarrow$ A

		y_1	…	y_T	Input Sequence
	o_1	o_2	…	o_N	All observations
s_1	B_{11}	B_{12}	…	B_{1N}	
s_2	B_{21}	B_{22}	…	B_{2N}	
…	…			…	
s_K	B_{K1}	B_{K2}	…	B_{KN}	

$\leftarrow$ B

Output Sequence (X) = $x_1, x_2, \ldots, x_T$

Named Entity Recognition

- Accurate recognition requires about 1M words of training data (1,500 news stories)
  - may be more expensive than developing rules for some applications
- Both rule-based and statistical approaches can achieve about 90% effectiveness for categories such as names, locations, organizations
  - others, such as product name, can be much worse

3. Document Structure and Markup

- Some parts of documents are more important than others
- Document parser recognizes structure using markup, such as HTML tags
  - Headers, anchor text, bolded text all likely to be important
  - Metadata can also be important
  - Links used for *link analysis*

Html Example

```
<html>
<head>
<meta name="keywords" content="Tropical fish, Airstone, Albinism, Algae eater,
Aquarium, Aquarium fish feeder, Aquarium furniture, Aquascaping, Bath treatment
(fishkeeping),Berlin Method, Biotope" />
…
<title>Tropical fish - Wikipedia, the free encyclopedia</title>
</head>
<body>
 …
<h1 class="firstHeading">Tropical fish</h1>
…
<p><b>Tropical fish</b> include <a href="/wiki/Fish" title="Fish">fish</a> found in <a
href="/wiki/Tropics" title="Tropics">tropical</a> environments around the world,
including both <a href="/wiki/Fresh_water" title="Fresh water">freshwater</a> and <a
href="/wiki/Sea_water" title="Sea water">salt water</a> species. <a
href="/wiki/Fishkeeping" title="Fishkeeping">Fishkeepers</a> often use the term
<i>tropical fish</i> to refer only those requiring fresh water, with saltwater tropical fish
referred to as <i><a href="/wiki/List_of_marine_aquarium_fish_species" title="List of
marine aquarium fish species">marine fish</a></i>.</p>
<p>Tropical fish are popular <a href="/wiki/Aquarium" title="Aquarium">aquarium</a>
fish , due to their often bright coloration. In freshwater fish, this coloration typically
derives from <a href="/wiki/Iridescence" title="Iridescence">iridescence</a>, while salt
water fish are generally <a href="/wiki/Pigment" title="Pigment">pigmented</a>.</p>
…
</body></html>
```


Professor Yuefeng Li
School of Computer Science

Html tags

- HTML files show more structure, where each field or element in HTML is indicated by a start tag (such as **<h1>**) and an optional end tag (e.g., **</h1>**).
- Elements can have attributes (with values), given by attribute_name = ”value” pairs.
- The **<head>** element contains metadata that is not displayed by a browser.
- The main heading is indicated by the <h1> tag , and terms that should be displayed in bold or italic are indicated by <b> or <i> tags etc.
- Links are commonly used, such as fish .

Tropical fish

From Wikipedia, the free encyclopedia

Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species. Fishkeepers often use the term *tropical fish* to refer only those requiring fresh water, with saltwater tropical fish referred to as *marine fish*.

Tropical fish are popular aquarium fish , due to their often bright coloration. In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

Hyperlinks

- Links are a key component of the Web.
- Important for navigation, but also for Web search
  - e.g., Example website
  - “Example website” is the anchor text
  - “http://example.com” is the destination link
  - both are used by search engines

Anchor Text

- Used as a description of the content of the *destination page*
  - i.e., collection of anchor text in all links pointing to a page used as an additional text field
- Anchor text tends to be short, descriptive, and similar to query text
- Retrieval experiments have shown that anchor text has significant impact on effectiveness for *some types of queries.*

References

[1] Chapter 4 in textbook - W. Bruce Croft, *Search Engines - Information retrieval in Practice*; Pearson, 2010.

[2] M. Albathan, **Y. Li**, Y. Xu, Using extended random set to find specific patterns, in Proceedings *of 2014 IEEE/WIC/ACM International Conference on Web Intelligence* (Vol. 2), 11–14 August 2014, Warsaw, Poland, 2014, pp. 30-37 (**Best Student Paper Award,** **https://dl.acm.org/citation.cfm?id=2682826**)