Topics for Today

- Text transformation
  - Word occurrence statistics
  - Tokenizing
  - Stopping and stemming
- Phrases
  - Document structure
  - Link analysis
  - Information extraction
  - Internationalization

Phrases

- 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”; “history book”; “deciduous trees”
  - Less ambiguous
    » e.g., “big apple” vs. “apple”
- Can be difficult to incorporate into ranking
  » 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 issues
  - Should phrases be indexed?
  - 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, for example
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: (a very old POS tagger)

Example Noun Phrases

<table>
<thead>
<tr>
<th>TREC data</th>
<th>Parent data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Phrase</td>
</tr>
<tr>
<td>69585</td>
<td>united states</td>
</tr>
<tr>
<td>61327</td>
<td>to</td>
</tr>
<tr>
<td>33864</td>
<td>los angeles</td>
</tr>
<tr>
<td>18062</td>
<td>hong kong</td>
</tr>
<tr>
<td>17758</td>
<td>beijing</td>
</tr>
<tr>
<td>17568</td>
<td>new york</td>
</tr>
<tr>
<td>15113</td>
<td>san diego</td>
</tr>
<tr>
<td>18095</td>
<td>orange county</td>
</tr>
<tr>
<td>12615</td>
<td>prime minister</td>
</tr>
<tr>
<td>12799</td>
<td>first time</td>
</tr>
<tr>
<td>12867</td>
<td>switzerland</td>
</tr>
<tr>
<td>10911</td>
<td>russian federation</td>
</tr>
<tr>
<td>9912</td>
<td>united nations</td>
</tr>
<tr>
<td>8127</td>
<td>southern california</td>
</tr>
<tr>
<td>7640</td>
<td>south korea</td>
</tr>
<tr>
<td>7620</td>
<td>south korea</td>
</tr>
<tr>
<td>7538</td>
<td>system</td>
</tr>
<tr>
<td>7524</td>
<td>system</td>
</tr>
<tr>
<td>7536</td>
<td>africa</td>
</tr>
<tr>
<td>7362</td>
<td>san francisco</td>
</tr>
<tr>
<td>7016</td>
<td>sarasota county</td>
</tr>
<tr>
<td>6792</td>
<td>city council</td>
</tr>
<tr>
<td>6348</td>
<td>middle east</td>
</tr>
<tr>
<td>6157</td>
<td>peace palace</td>
</tr>
<tr>
<td>5955</td>
<td>human rights</td>
</tr>
<tr>
<td>5837</td>
<td>white house</td>
</tr>
</tbody>
</table>

N-Grams

- Frequent n-grams are more likely to be meaningful phrases
- N-grams form a Zipf distribution
  - Better fit a Zipf distribution 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 $2 \leq n \leq 5$

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
Google N-Grams

- Web search engines index n-grams
- Google n-gram 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”
  - In Chinese, “limited liability corporation”

Google books n-gram corpus

- Automatically extract structure from text
  - annotate document using tags to identify extracted structure

- Named entity recognition
  - identify phrases that refer to something of interest in a particular application
  - e.g., people, companies, locations, dates, product names, prices, etc.

Topics for Today

- Text transformation
  - Word occurrence statistics
  - Tokenizing
  - Stopping and stemming
  - Phrases
  - Document structure
  - Link analysis
  - Information extraction
  - Internationalization

Information Extraction
Named Entity Recognition

Example showing semantic annotation of text using XML tags

Information extraction also includes document structure and more complex features such as relationships and events

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

Rules either developed manually by trial and error or using machine learning techniques

Statistical

- use 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

Covered in CS4740 NLP

HMM for Extraction

- Resolve ambiguity in a word using context
  - 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
Learn a HMM from training data

• Each state is associated with a probability distribution over words (the output)

HMM for Extraction

- 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

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 can achieve about 90% effectiveness for categories such as names, locations, organizations
  - others, such as product name, can be much worse

NE recognition

- Not generally found to be helpful during search
- Useful for domain-specific or vertical search engines
- Useful for displaying results, browsing
- Critical for question answering applications
Internationalization

- 2/3 of the Web is in English
- About 50% of Web users do not use English as their primary language
- Many (maybe most) search applications have to deal with multiple languages
  - monolingual search: search in one language, but with many possible languages
  - cross-language search: search in multiple languages at the same time

Internationalization

- Many aspects of search engines are language-neutral
- Major differences:
  - Text encoding (converting to Unicode)
  - Tokenizing (many languages have no word separators)
  - Stemming
- Cultural differences may also impact interface design and features provided

Chinese “Tokenizing”

1. Original text
   早灾在中国造成的影响
   (the impact of droughts in China)

2. Word segmentation
   旱灾 在 中 国 造成 的 影响
   drought at china make impact

3. Bigrams
   旱灾 灾在 在 中 国 造 成 的 影 影 响