Class Logistics

Demo slots
- Four groups already signed up, first come first serve, grab your spot early!
- Check on course homepage for opening slots

Project
- (Optional) email the instructors a one-page proposal, we will take a look and give you feedback.

11/7 deadline

Questions on Homework 2?
- No extension, but ...
- For Doconly, we will only run Rankers that it supports

Midterm Exam (10/31, due 11/07)
- Cover material from all the lectures including lecture 6 on 10/31
- Individual take-home exam, do not discuss with others at all
  - This is serious, do it on your own
  - People have been caught in previous semesters
Review of Last Lecture

Inverted Index

Index Compression and Construction

Query Processing

• Data structure: document-only, document + word occurrences

• Access methods: query-level and term-level

Index Compression and Construction

• Elias codes

• Byte-aligned code

• Delta encoding, skip pointer

<table>
<thead>
<tr>
<th>k</th>
<th>Binary Code</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00000001</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>00000010</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>00000111</td>
<td>03</td>
</tr>
<tr>
<td>4</td>
<td>00001110</td>
<td>04</td>
</tr>
<tr>
<td>5</td>
<td>00011111</td>
<td>05</td>
</tr>
<tr>
<td>6</td>
<td>00111111</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>11111111</td>
<td>07</td>
</tr>
<tr>
<td>8</td>
<td>11111111</td>
<td>08</td>
</tr>
<tr>
<td>9</td>
<td>11111111</td>
<td>09</td>
</tr>
<tr>
<td>10</td>
<td>11111111</td>
<td>0A</td>
</tr>
</tbody>
</table>

Elias (1,2,(2,2)), (1,3,(7,11,5)), (1,2,(2,4)), (1,2,(3,10))
Review of Last Lecture
Query Processing and Optimization
- Ignoring documents and/or terms
- Reordering and bucketization

Architecture

Document Processing (Text Processing)
A core component of Information Retrieval systems
- Crawling is specific to Web search
Converting raw documents into terms to be indexed
What’s the main objective?
- Enabling the matching of terms in the query to those in the documents.
Document processing and query parsing are connected.
Key Issues in Document Processing

Encoding characters correctly and across languages

Segmenting sentences into words and phrases

Token pruning based on corpus characteristics

Simple normalization
  • Punctuation (them?, die), capitalization (Whether), abbreviation (I.B.M.)

Advanced normalization
  • Stemming, misspelling, synonyms and related words
Character Encoding

A character encoding is a mapping between bits and characters.

- ASCII: 100 0001 → A;
- Unicode UTF-8: 11100111 10001000 10110001 → 爱

ASCII was standardized in 1963, for English letters, numbers, common punctuations, and control characters.

- 7-bits, an additional 1 bit can be used for error checking [how?]
- \(2^7 = 128\) characters

128 characters is far from enough for many languages!

- Chinese: 50,000+ characters, and ~3,000 commonly used

Unicode

Before Unicode, it was a mess!

A universal mapping from numbers to characters for all (past, present, future) languages

- Unicode does not specify bits, only codepoints (in hex)
  - \(U+7231 = \text{ancock}\)
- Unicode has the capacity to encode millions of characters

To actually use Unicode, a specific character encoding is required to encode the codepoint

- UTF-32: fixed length, each codepoint takes 4 bytes
- UTF-8: variable length, each codepoint takes 1-4 byte(s)

UTF-8

If the codepoint is less than or equal to 127, i.e., ASCII

- Use a single byte, with leading 0

If the codepoint is greater than 127

- In the first byte, most significant bits are used to indicate how many bytes are in the codepoint in unary fashion, with a separator 0 bit
- For subsequent bytes, use 10 as the most significant bits
  - Why use 10 at all?
- The rest of bits together encode the codepoint.

Example: \(11100111 10001000 10110001 \rightarrow 0111\ 0010\ 0011\ 0001\ = 0x7231\)
In-Class Exercise

Decode the following bits into unicode characters via UTF-8

11000111 10000101 00001011
11000111 10000101 00001011 U+01C5
00001011 00001011 U+000B

UTF-8

How many characters can UTF-8 encode?
• 4 bytes with 11 control bits for a total of 21 bits, 2+ millions
• What about UTF-32?

UTF-8 vs UTF-32
• UTF-8 is compatible with ASCII and uses less space (great for storage)
• UTF-32 allows for faster lookup, finding the nth character in a byte array is as simple as computing \( 4 \times n \) (good for in memory access)

Key Issues in Document Processing

Encoding characters correctly and across languages
Segmenting sentences into words and phrases
Token pruning based on corpus characteristics
Simple normalization
Advanced normalization
Segmmentation / Tokenization

Western languages, such as English and French
• Words are separated by spaces: New York University or Université de New York

CJK languages (Chinese, Japanese, Korean)
• Words are not separated: 新约大学 or ニューヨーク大学
• 大学: University != 大 学: study
• Imagine in English: newyorkuniversity …

Segmentation is critically important, and, surprisingly, not just limited to CJK languages. Any example?
• URLs don’t have spaces!
  • whitehouse.gov; handynasty.net

Statistical Segmentation

Morphological analysis and dictionary based approach
• Can not handle emerging words and unknown words

 Mostly-unsupervised statistical segmentation of Japanese, by Ando and Lee, NAACL 2000

Intuition: Given a large corpus, character sequences that form meaningful words are expected to be more frequently together than those that are not.
• newyorkuniversity: new|york|university vs newy|orkuni|versity

Statistical Segmentation

N-gram model
• Split the sequence into successive words of $n$ characters each
  • E.g., $n = 4$: ABCDEFGH $\rightarrow$ ABCD, BCDE, CDEF, DEFG, EFGH

Step 1: For each location, ask the following for all six $(i, j)$ pairs:
• In the corpus, is $|s_i| > |t_j|$?
• Yes is a support for the location being a correct word boundary.

$\begin{array}{c}
\text{Step 1: For each location, ask the following for all six ($i, j$) pairs:} \\
\text{• In the corpus, is } |s_i| > |t_j|? \\
\text{• Yes is a support for the location being a correct word boundary.}
\end{array}$
Statistical Segmentation

Step 2: Normalize based on \( n \)
- \( n = 2, 2 \) questions; \( n = 3, 4 \) questions; \( n = 4, 6 \) questions

\[
v_n(k) = \frac{1}{2(n - 1)} \sum_{i=1}^{2} \sum_{j=1}^{n-1} I_{>}(s_i^j, \#(t_i^j))
\]

Step 3: Normalize across all \( n \)

\[
v_N(k) = \frac{1}{|N|} \sum_{n \in N} v_n(k)
\]

Statistical Segmentation

Step 4: Identify boundaries \( l \), if:
- \( v_n(l) = v_n(l - 1) \& v_n(l) > v_n(l + 1) \); local maximum
- \( v_n(l) > t \); allow for single character words

In-Class Exercise

where is the word boundary (using \( n=2 \))? 

Google Search:

\(^1\) 49M results
\(^2\) 4.9M results
\(^3\) 22K results
\(^4\) 694K results
\(^5\) 43M results

\(^1\) | \(^2\) | \(^3\) | \(^4\) | \(^5\) |
1 2 0 2 6  =>  \(^1\) | \(^2\) | 0 | \(^4\) | \(^5\) |
Key Issues in Document Processing

- Encoding characters correctly and across languages
- Segmenting sentences into words and phrases
- Token pruning based on corpus characteristics
- Simple normalization
- Advanced normalization

Text Statistics

Distribution of word frequencies is very skewed (AP89 data)

Zipf’s Law

- The product of the rank of a word and its frequency is a constant:
  \[ r(w) \times f(w) = k \text{ or } r(w) \times \Pr(w) = c \]
- For English \( c \approx 0.1 \)
Stopwords
With $c \approx 0.1$, the most frequent word will account for close to 10% of all word occurrences!
- the appears 6%
- Those very frequent words are called stopwords
- They can be corpus dependent: e.g., if your corpus has articles only from wikipedia, the word wiki may become a stopword

The postings lists for stopwords are very long
- They are very costly for query processing
- They often do not contribute much to the final ranking

However:
- "to be or not to be that is the ..." -- Hamlet Act 3 Scene 1

In modern search engines, stopwords are indexed, but ignored for most queries, only used for certain queries.

Normalization
The main objective of document parsing?
- Matching query terms to document terms

Users are lazy:
- "New York University" ? “new york university”
- “U.S.A. population” ? "usa population”
- “What’s the highest mountain?” ? “what’s the highest mountain”

The goal of normalization is to have a canonical representation of the terms so that document terms and query terms can be matched.

Simple Normalization
Punctuation
- Leading and trailing punctuations are in general separated from the term
- Symbols like “-” are usually used as separators

Abbreviations
- Simple abbreviations are transformed into single words:
  - I.B.M. → ibm
- More complex ones are handled differently

Capitalization
- Terms are usually lowercased
- “US” and “us” are different, but hopefully the surrounding context will compensate for this ambiguity.
Simple Normalization

Those transformations must be applied on queries
- Copy-and-paste queries
- Some punctuations, such as `.`, appear frequently in queries
- Without transformation on the query side as well, they won’t match the document terms.

All above transformation must be tracked, why?
- Snippets returned to the user must reflect the actual document content for readability and satisfying users’ expectation

Key Issues in Document Processing

Encoding characters correctly and across languages

Segmenting sentences into words and phrases

Token pruning based on corpus characteristics

Simple normalization

Advanced normalization
- Stemming, misspelling, synonyms and related words

Some Examples

Words can be written differently due to singular/plural forms or past/present/future tenses
- ran, running ➔ run
- universities ➔ university

Words can often be misspelled
- marshmellow ➔ marshmallow
- miniture ➔ miniature

Words can have synonyms and related words
- university ➔ college, school
- dog ➔ canine, pet
Stemming

The process of finding the **semantic root** of a word

- Crucial for highly inflected languages such as Arabic
- Some Turkish verbs can have 100’s of inflected forms

For example, the word “write” in Arabic:

<table>
<thead>
<tr>
<th>Active</th>
<th>Passive</th>
</tr>
</thead>
<tbody>
<tr>
<td>past</td>
<td>present</td>
</tr>
<tr>
<td>подарил</td>
<td>написал</td>
</tr>
<tr>
<td>давал</td>
<td>писал</td>
</tr>
<tr>
<td>написал</td>
<td>писал</td>
</tr>
<tr>
<td>писал</td>
<td></td>
</tr>
</tbody>
</table>

Approaches for Stemming

Dictionary-based

Algorithmic

- Suffix-stripping
  - Stem the word based on the suffix it has

- Lemmatisation
  - Stem the word in the context of the sentence it appears in

- Machine Learning / Stochastic
  - Train a probabilistic model from a set of known roots and their different inflection forms

Porter Stemming Algorithm


Most widely used stemming algorithm for English

A suffix-stripping algorithm

Open source version of the algorithm readily available in many programming languages on the Web.
Overview of the Porter Algorithm

- Step 1a: removes plurals
- Step 1b: removes –ed(ly) or –ing(ly) suffixes
- Step 1c: turns –y to –i
- Step 2: handles double suffixes such as –ization
- Step 3: handles –full, –ness, etc.
- Step 4: handles –ant, –ence, etc.
- Step 5: removes final –e and –ll

Comments on the Porter Algorithm

The stem may not be a full word, in fact it is often not

- vegetables → veget
- natural → natur

The algorithm is quite aggressive, for example:

- organization → organ
- orienteering → orient

A Porter2 algorithm has been developed to address some of the common errors

- [http://snowball.tartarus.org/](http://snowball.tartarus.org/)
- It also provides a way to specify exceptions so that known errors can be corrected easily.

Stemming in Web Search

Aggressive stemming is not desirable in practice.

Conflating two different words, such as orienteering and oriental, can cause major confusion to the users. Why?

- Users are often very upset if irrelevant results are returned for no perceivable reason
- Imagine a user who searches for "military organization" and sees organ donation results instead...
**Spelling Correction**

Words in documents can also be misspelled

- Queries are even more likely to be misspelled, 10%+

A spelling dictionary is traditionally used to correct typos

- Using edit distance to measure a word with a known word in the dictionary
- Using n-gram or soundex to speed up the lookup process

**Edit Distance**

Given two strings, s and t, the edit distance, or Levenshtein Distance, between them is the minimum number of edit operations required to transform s into t.

Edit operations

- Insertion
- Deletion
- Replacement

For example: edit-distance("cats", "fast") = 3

- c \rightarrow f, t \rightarrow s, s \rightarrow t; or
- c \rightarrow f, insert s before t, delete s after t

**Computing Edit Distance: Dynamic Programming**

\[
\begin{align*}
\text{lev}_{ab}(i, j) & = \min \left\{ \text{lev}_{ab}(i-1, j) + 1, \text{lev}_{ab}(i, j-1) + 1, \text{lev}_{ab}(i-1, j-1) + [a_i \neq b_j] \right\}
\end{align*}
\]
In-Class Exercise

Compute edit-distance("trials", "trickle")

<table>
<thead>
<tr>
<th></th>
<th>t</th>
<th>r</th>
<th>i</th>
<th>c</th>
<th>k</th>
<th>l</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Insert c, replace a with k
Replace a with c, insert k

Indexing Support for Computing Edit Distances

The spelling dictionary can be quite large
• Edit distance computation is $O(|s| \times |t|)$, running it against every word in the dictionary is very expensive
• Only interested in words with small edit distance to our word

N-gram indexing
• Looks for words containing a minimum number of n-grams from the given word/query

Soundex indexing
• Looks for words pronounced the same as the given word/query

N-Gram Indexing

Example: bord $\Rightarrow$ bo, or, rd
• Quickly identify words with at least two n-grams among the three, how?
• Using n-gram based postings lists!
N-Gram Indexing

Furthermore, not all the words from the index must be compared using the full dynamic programming algorithm

- edit-distance(bord, boardroom) ≥ 5, why?
- length(bord) = 4 and length(boardroom) = 9

Soundex

- marshmallow → M625
  - Step 1: Marshmallow
  - Step 2: M-rs-m-
  - Step 3: M-62-5-44-
  - Step 4: M-62-5-4-
  - Step 5: M6254
  - Step 6: M625

- marshmellow → M625
  - Keep the first letter (in upper case).
  - Replace these letters with hyphens: a,e,i,o,u,y,h,w.
  - Replace the other letters by numbers as follows:
    - b,f,p,v
    - c,g,j,q,x,z
    - d, t
    - k, m,n
    - s
  - Delete adjacent repeats of a number
  - Delete the hyphens.
  - Keep the first three numbers or pad out with zeros.

Soundex

- Only words with the same soundex code are retrieved from the hash-based index

- Again, some word length based pruning can apply
Spelling Checking in Web Search

Similar to stemming, the system needs to be very careful not to produce unexpected results.

Rarely performed during document parsing
- Very expensive to perform on all words on all the pages

Very important in query parsing!
- Users have become increasingly relied on search engine as the spell checker!

Synonyms and Related Words

Much harder problems than stemming and spell checking

Traditional IR approaches rely on looking up in manually generated knowledge bases.

WordNet 3.0
- Available online and used by numerous IR systems
  - http://wordnet.princeton.edu/wordnet/
  - 155K words organized into 117K synsets
- George Miller, a psychologist, started the project in late 1980’s
- A comprehensive lexical database / thesaurus for English
  - Relationships between words are labeled

Important Relationships Between Words

Synonymy
- college ~ university

ISA relationship
- dreamliner ISA plane

Both Synonymy and ISA relationships can be consulted to find synonyms and related words

Is-Part-Of relationship
- Instagram Is-Part-Of Facebook

Antonymy
- young vs old
Limitations of IR Approaches

Driven by manually constructed knowledge

• Spelling dictionary, WordNet, etc.
• Impossible to be comprehensive for all/emerging words

Inherently subjective bias

• E.g., WordNet is maintained by a small number of people

Not extendable to other languages

Statistical Approaches

Similar to statistical segmentation.

General approach:

• Decide on the source corpus: Web pages, query logs, etc.
• Adopt a computation window: sentence, document, query, query session
• Compute the individual frequencies of words for the windows
• Compute the co-occurrences of word pairs for the windows

Measure the degree of association based on various formulas
Measures (from Lecture 3)

Dice’s Coefficient: proportions of two words’ occurrences that are co-occurrences

\[
\frac{2 \cdot n_{ab}}{n_a + n_b} \frac{\text{rank}}{n_a + n_b}
\]

Pointwise Mutual Information: using probabilities to measure the degree to which both words occur independently

\[
\log \frac{P(a,b)}{P(a)P(b)} \frac{\text{rank}}{n_a \cdot n_b}
\]

More Measures

Example Statistical Approach

Data Source: User query logs

Window: Query session

- A sequence of queries issued by the same user to fulfill his information needs

A user tends to issue another similar, and slightly modified, query if his first query does not get satisfactory results.

As a result, words co-occur frequently in the same session, but in different queries are likely to be synonyms or somehow related.
Machine Learning Approach: word2vec

Efficient Estimation of Word Representations in Vector Space, by Mikolov, Chen, Corrado, Dean, 2013

Intuition: similar words have similar context

As usual, compute the conditional probabilities using the statistical approach between word pairs

Treat those word pairs as training examples to train a neural network language model

- Where words are represented as vectors
- Where the model maximizes the likelihood of the observed probabilities in the source corpus.

Surprising benefit: the vector space preserves relationships

Normalization in Web Search

All three advanced normalization tasks (stemming, spell correction, synonym & related words) are cautiously used in the document processing stage

- It is often too dangerous to alter the original words too aggressively
- If a mistake is made, it will affect all queries and can not be fixed until the index is rebuilt

Instead, those normalization tasks are typically applied to the queries through *query rewriting and expansion*

- More flexible and easy to experiment
- Mistakes can be corrected fast
Query Rewriting and Expansion

Goal: Modify the user query to produce alternative and/or additional terms that can be used to retrieve more and better documents.

A rich research field with many technical papers from both academia and industry.

In addition to those advanced normalization tasks
• Grouping terms into phrases
• Mapping terms to canonical semantic entities
• and many more …

Key material of the Lecture

Character encoding (UTF-8)

Segmentation (statistical segmentation)

Normalization (stemming, spelling correction, related words)