Administrative

- Assignment 2
- Assignment 1
  - Overall, pretty good
  - Hard to get right!
  - Write-up:
    - be clear and concise
    - think about the point(s) that you want to make
    - justify your answer
- hw 2 back soon...

Quick recap

If we have a dictionary, with postings lists containing weights (e.g. tf-idf) explain briefly (e.g. pseudo-code) how to calculate the document similarities between a query of two words

Name two speed challenges that are faced when doing ranked retrieval vs. boolean retrieval.

One way to speed up ranked retrieval is to only perform the full ranking on a subset of the documents (inexact K). Name one method for selecting this subset of documents
So far…

query → IR system → ranked docs → what are we missing?

Today

User interface/user experience:

Once the documents are returned, how do we display them to the user?

Middlebury college
(spelling correction)

www.fordvehicles.com/cars/mustang/
en.wikipedia.org/wiki/Ford_Mustang
www.mustangseats.com/
www.mustangsurvival.com/

How is this?
In many domains, we have document metadata
web pages: titles, URLs, …
academic articles: what information do we have?

Other times, we may not have explicit meta-data, but may still want to provide additional data
- Web pages don't provide “snippets”/summaries

Even when pages do provide metadata, we may want to ignore this. Why?

The search engine may have different goals/motives than the webmasters, e.g. ads

quality you can trust at a price

you can afford. shop small!

Los Angeles, CA
Summaries

We can generate these ourselves!

Most common (and successful) approach is to extract segments from the documents (called extractive in contrast with abstractive).

How might we identify good segments?
- Text early on in a document
- First/last sentence in a document, paragraph
- Text formatting (e.g. <h1>)
- Document frequency
- Distribution in document
- Grammatical correctness
- User query!

Summaries

Simplest heuristic: the first X words of the document

More sophisticated: extract from each document a set of “key” sentences
- Use heuristics to score each sentence
- Learning approach based on training data
- Summary is made up of top-scoring sentences

Segment identification

extract features

learning approach

segment identifier

hand-label “good” segments/sentences

Summaries

A static summary of a document is always the same, regardless of the query that hit the doc

A dynamic summary is a query-dependent attempt to explain why the document was retrieved for the query at hand

Which do most search engines use?
Summaries

Dynamic summaries

Present one or more “windows” within the document that contain several of the query terms
- “KWIC” snippets: Keyword in Context presentation
- Generated in conjunction with scoring
  - If query found as a phrase, all or some occurrences of the phrase in the document
  - If not, document windows that contain multiple query terms
- The summary gives the entire content of the window – all terms, not only the query terms

Dynamic vs. Static

What are the benefits and challenges of each approach?

Static
- Create the summaries during indexing
- Don’t need to store the documents

Dynamic
- Better user experience
- Makes the summarization process easier
- Must generate summaries on the fly and so must store documents and retrieve documents for every query!

Generating dynamic summaries

If we cache the documents at index time, can find windows in it, cueing from hits found in the positional index
- E.g., positional index says “the query is a phrase in position 4378” so we go to this position in the cached document and stream out the content
- Most often, cache only a fixed-size prefix of the doc

Note: Cached copy can be outdated!
Dynamic summaries

Producing good dynamic summaries is a tricky optimization problem:

- The real estate for the summary is normally small and fixed
- Want short item, so show as many KWIC matches as possible, and perhaps other things like title

Users really like snippets, even if they complicate IR system design

Alternative results presentations?

An active area of HCI research
An alternative: http://www.searchme.com/ copies the idea of Apple's Cover Flow for search results
Spelling correction

How might we utilize spelling correction?
Two common uses:
- Correcting user queries to retrieve “right” answers
- Correcting documents being indexed

Document correction

Especially needed for OCR’ed documents
- Correction algorithms are tuned for this
- Can use domain-specific knowledge
  - E.g., OCR can confuse O and D more often than it would confuse O and I (adjacent on the keyboard)

Web pages and even printed material have typos

Often we don’t change the documents but aim to fix the query-document mapping

Query misspellings

Our principal focus here
- e.g., the query Alanis Morisett

What should/can we do?
- Retrieve documents indexed by the correct spelling
- Return several suggested alternative queries with the correct spelling
  - Did you mean ... ?
- Return results for the incorrect spelling
- Some combination

Advantages/disadvantages?
Spelling correction

Two main flavors/approaches:

Isolated word: Check each word on its own for misspelling
Which of these is misspelled?
- motor
- from
Will not catch typos resulting in correctly spelled words

Context-sensitive
- Look at surrounding words,
- E.g., I flew from Heathrow to Narita.

Isolated word correction

Fundamental premise – there is a lexicon from which the correct spellings come

Choices for lexicon?
- A standard lexicon such as Webster’s English Dictionary
- An “industry-specific” lexicon – hand-maintained
- The lexicon of the indexed corpus
  - E.g., all words on the web
  - All names, acronyms etc.
  - (Including the misspellings)

Isolated word correction

Given a lexicon and a character sequence Q, return the words in the lexicon closest to Q

Lexicon

\[ q_1, q_2, \ldots, q_m \]

How might we measure “closest”?

Edit distance

Given two strings \( S_1 \) and \( S_2 \), the minimum number of operations to convert one to the other

Operations are typically character-level
- Insert, Delete, Replace, (Transposition)

E.g., the edit distance from \( dog \) to \( dot \) is \( 1 \)
- from \( cat \) to \( act \) is \( 1 \) (with transpose?)
- from \( cat \) to \( dog \) is \( ? \)

Generally found using dynamic programming

What’s the problem with basic edit distance?
**Weighted edit distance**

Not all operations are equally likely!

Character-specific weights for each operation
- OCR or keyboard errors, e.g., \( m \) more likely to be mistyped as \( n \) than as \( q \)
- replacing \( m \) by \( n \) is a smaller edit distance than by \( q \)
- This may be formulated as a probability model

Requires weight matrix as input
Modify dynamic programming to handle weights

---

**Using edit distance**

We have a function \( \text{edit} \) that calculates the edit distance between two strings

We have a query word
We have a lexicon

\[ q_1 q_2 \ldots q_m \]

Now what?

---

**Enumerating candidate strings**

Given query, enumerate all character sequences within a preset (weighted) edit distance (e.g., 2)

\( \text{dog} \quad \text{doa, dob, …, do, og, …, dogs, dogm, …} \)

Intersect this set with the lexicon

---

**Using edit distance**

We have a function \( \text{edit} \) that calculates the edit distance between two strings

We have a query word
We have a lexicon

\[ q_1 q_2 \ldots q_m \]

Naive approach is too expensive!
Ideas?
Just like word n-grams, we can talk about character n-grams.

A character n-gram is \( n \) contiguous characters in a word.

<table>
<thead>
<tr>
<th>unigrams</th>
<th>bigrams</th>
<th>trigrams</th>
<th>4-grams</th>
</tr>
</thead>
<tbody>
<tr>
<td>remote</td>
<td>remot</td>
<td>remot</td>
<td>remo</td>
</tr>
<tr>
<td>reemote</td>
<td>reemote</td>
<td>remo</td>
<td>remo</td>
</tr>
</tbody>
</table>

Two challenges: quantifying overlap and speed!

What is the trigram overlap between “november” and “december”?

3 trigrams of 6 overlap. How can we quantify this?
Correct proportion?

Overlap = 3/6

Any problems with this?

Correct proportion?

Overlap = 3/6

Ignores number of n-grams in the candidate word

Other ideas?
One option – Jaccard coefficient

Let $X$ and $Y$ be two sets; then the J.C. is

$$\frac{|X \cap Y|}{|X \cup Y|}$$

What does this mean?

- $|X \cap Y|$ number of overlapping n-grams
- $|X \cup Y|$ total n-grams between the two

Example

<table>
<thead>
<tr>
<th>November</th>
<th>December</th>
</tr>
</thead>
<tbody>
<tr>
<td>nov</td>
<td>dec</td>
</tr>
<tr>
<td>ove</td>
<td>ece</td>
</tr>
<tr>
<td>vem</td>
<td>cem</td>
</tr>
<tr>
<td>emb</td>
<td>emb</td>
</tr>
<tr>
<td>mbe</td>
<td>mbe</td>
</tr>
<tr>
<td>ber</td>
<td>ber</td>
</tr>
</tbody>
</table>

$|X \cap Y| = 3$  \quad JC = 1/3  

$|X \cup Y| = 9$

Jaccard coefficient

Equals 1 when $X$ and $Y$ have the same elements and zero when they are disjoint

- $X$ and $Y$ don’t have to be of the same size
- Always assigns a number between 0 and 1
- Threshold to decide if you have a match
  - E.g., if J.C. $> 0.8$, declare a match

Efficiency

We have all the n-grams for our query word

How can we efficiently compute the words in our lexicon that have non-zero n-gram overlap with our query word?
**Efficiency**

We have all the n-grams for our query word

How can we efficiently compute the words in our lexicon that have non-zero n-gram overlap with our query word?

Index the words by n-grams!

| lo  | alone | lord | sloth |

**Matching trigrams**

Consider the query **lord** – we wish to identify words matching 2 of its 3 bigrams (**lo**, **or**, **rd**)

| lo  | alone | lord | sloth |
| or  | border | lord | morbid |
| rd  | ardent | border | card |

**Context-sensitive spell correction**

Text: *I flew from Heathrow to Narita.*

Consider the phrase query "**flew from Heathrow**"

We’d like to respond: Did you mean "**flew from Heathrow**"?

How might you do this?

**Context-sensitive correction**

Similar to isolated correction, but incorporate surrounding context

Retrieve dictionary terms close to each query term (e.g. isolated spelling correction)

Try all possible resulting phrases with one word “fixed” at a time

- flew from heathrow
- fled form heathrow
- flea form heathrow

Rank alternatives based on frequency in corpus

Can we do this efficiently?
Another approach?

What do you think the search engines actually do?

Often a combined approach

Generally, context-sensitive correction

One overlooked resource so far...

---

 Query logs

<table>
<thead>
<tr>
<th>AnonID</th>
<th>Query</th>
<th>QueryTime</th>
<th>ItemRank</th>
<th>ClickURL</th>
</tr>
</thead>
<tbody>
<tr>
<td>252440</td>
<td>evergreen real estate co.</td>
<td>2006-05-19 06:32:42</td>
<td>6</td>
<td><a href="http://www.eraevergreen.com">http://www.eraevergreen.com</a></td>
</tr>
</tbody>
</table>

How might we use query logs to assist in spelling correction?

---

Query logs

Find similar queries
“flew form heathrow” and “flew from heathrow”

Query logs contain a temporal component!

<table>
<thead>
<tr>
<th>term</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>osgood schlatter</td>
<td>1</td>
</tr>
</tbody>
</table>

1 result (0.17 seconds)

Attempt 1: one doc retrieved, don’t click on any docs

---

Query logs

Find similar queries
“flew form heathrow” and “flew from heathrow”

Query logs contain a temporal component!

<table>
<thead>
<tr>
<th>term</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>westgate vacation villas</td>
<td>2</td>
</tr>
</tbody>
</table>

1 result (0.17 seconds)

Attempt 2: may docs retrieved click on one doc, but quickly issue another query
Query logs

Find similar queries
“flew from heathrow” and “flew from heathrow”

Query logs contain a temporal component!

General issues in spell correction

Do we enumerate multiple alternatives for “Did you mean?”

Need to figure out which to present to the user

Use heuristics
- The alternative hitting most docs
- Query log analysis + heuristics
  - For especially popular, topical queries

Spell-correction is computationally expensive
- Avoid running routinely on every query?
- Run only on queries that matched few docs