Vocabulary mismatch

Query: U.S. ends Gaddafi operation

Document vector

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Query vector

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Represent vector as a hitlist:

\[ x(D) = \sum_{i=1}^{10} \frac{y_i}{\sqrt{v_i^2 + \epsilon}} \]

Tokenizing text

- Tokenizer: characters → tokens ("words")
  - token: atomic indexing unit; "end" will not match "ends"
  - token: any contiguous sequence of letters / numbers
  - how to handle other symbols?
    - periods: I.B.M., Ph.D.
    - hyphens: Spanish-speaking, pre-diabetes, e.g.
    - apostrophes: O'connell, 1980's, men's
  - Split: I.B.M. → I, B, "M"
  - accentual matches: pronoun "I" will match I.B.M.
  - Fuse: "pre-diabetes" → "pre-diabetes"
  - mismatch: "diabetes" will not match "pre-diabetes"

Tokenizing text (2)

- If proximity indexing supported:
  - split, store positions: I.B.M. → I, B, "M"
  - use proximity queries: #"I"("B", "M")

- If proximity not supported:
  - both split and fuse: I.B.M. → I, B, "M", IBM
  - same spelling in document and query → 4 matches
  - different spellings → partial match: I → 1 match
  - may seem like adding noise: misconception
  - different matches will have different weights

- Critical: apply same rules to docs and queries

Tokenizing Asian text

- Words = natural units in western languages
- Asian text = continuous stream of characters
- no explicit word separators
- meaning in sub-sequences

Learn segmentation

- supervised: need examples
- unsupervised: spot common sequences (compression)

Half-overlapping n-grams
- mix over different n

Morphological Variation

- Need to recognize various forms of the same word
  - policy, policies, polices, politician... all have similar meaning
  - but not the same as: police, policeman, policemen
  - if query contains "political", which forms should it match?

- Highly-influential languages (e.g. Arabic)
  - several thousand "roots" (core concepts)
  - e.g. the root "kādūb" - carries the meaning "writing"
  - actual words formed by inserting syllables before (prefix), after (suffix) and inside ( infix)
  - fine-grained distinctions based on context
  - about 20 different ways to spell the word "Arab"

Stemming

- Stemmer: reduce word forms to proper class
- Porter: rule-based stemmer
- sequence of suffix-stripping rules
- produces "stems", makes a number of mistakes
- http://snowball.tartarus.org/
- Krovetz: algorithmic + dictionary
- dictionary lookup
- if not found → strip suffix, lookup again
- produces "words" not stems
- lower FP (policy→police, higher FN (noise/hoisy)

Porter Stemmer (step 1 of 5)

Step 1a:
- Replace zero by as (e.g., stresss → stress)
- Delete if the preceding word part contains a vowel not immediately before the r (e.g., gaps → gap but gap → gaps)
- Replace if ed or es by if preceded by more than one letter, otherwise by ed (e.g., thin → thinned, cute → cut)
- If suffix is us or as do nothing (e.g., stresss → stress)

Step 1b:
- Replace end, ends by ed or es if it is the last part of the word after the first non-stop ending (e.g., agreed → again, found → found)
- Delete ed, es before if the preceding word part contains a vowel, and then if the word ends in at, ed, or es add e (e.g., fitted → fit, pitting → pit, sitting → sit), or if the word ends with a double letter that is not ht, th, or th as, remove the last letter (e.g., falling → fall, topping → top), or if the word is short, add e (e.g., hugging → hug)
- Whew!
Character n-grams

- Building a good stemmer is hard
- Cheap alternative:
  - take every n-character substring of the word
  - related words \( \Rightarrow \) many of the same n-grams
- n=4,5 works well for European languages

Stemming vs. character N-grams

<table>
<thead>
<tr>
<th>Language</th>
<th># Days</th>
<th>Words</th>
<th>Morfessor</th>
<th>Snowball</th>
<th>4-grams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>85,527</td>
<td>0.2195</td>
<td>0.2786 (26.9%)</td>
<td>0.3163 (44.1%)</td>
<td></td>
</tr>
<tr>
<td>Czech</td>
<td>61,735</td>
<td>0.2175</td>
<td>0.3215 (14.2%)</td>
<td>0.3204 (45.1%)</td>
<td></td>
</tr>
<tr>
<td>Danish</td>
<td>190,003</td>
<td>0.4652</td>
<td>0.6274 (27.7%)</td>
<td>0.4273 (72.7%)</td>
<td></td>
</tr>
<tr>
<td>Dutch</td>
<td>166,174</td>
<td>0.4829</td>
<td>0.6165 (11.7%)</td>
<td>0.5095 (14.3%)</td>
<td></td>
</tr>
<tr>
<td>English</td>
<td>25,244</td>
<td>0.3191</td>
<td>0.3964 (18.5%)</td>
<td>0.4174 (17.4%)</td>
<td></td>
</tr>
<tr>
<td>French</td>
<td>129,804</td>
<td>0.3287</td>
<td>0.4231 (13.6%)</td>
<td>0.4153 (43.1%)</td>
<td></td>
</tr>
<tr>
<td>German</td>
<td>294,805</td>
<td>0.3489</td>
<td>0.4222 (15.3%)</td>
<td>0.3822 (17.4%)</td>
<td></td>
</tr>
<tr>
<td>Hungarian</td>
<td>253,593</td>
<td>0.3979</td>
<td>0.6202 (48.2%)</td>
<td>0.5599 (67.9%)</td>
<td></td>
</tr>
<tr>
<td>Italian</td>
<td>175,158</td>
<td>0.3951</td>
<td>0.4350 (18.2%)</td>
<td>0.3925 (65.2%)</td>
<td></td>
</tr>
<tr>
<td>Portuguese</td>
<td>210,754</td>
<td>0.5332</td>
<td>0.5301 (5.5%)</td>
<td>0.3316 (4.2%)</td>
<td></td>
</tr>
<tr>
<td>Russian</td>
<td>16,712</td>
<td>0.3071</td>
<td>0.3307 (23.8%)</td>
<td>0.3406 (27.5%)</td>
<td></td>
</tr>
<tr>
<td>Spanish</td>
<td>454,041</td>
<td>0.4265</td>
<td>0.6320 (18.2%)</td>
<td>0.4689 (24.7%)</td>
<td></td>
</tr>
<tr>
<td>Swedish</td>
<td>142,819</td>
<td>0.3387</td>
<td>0.3738 (10.4%)</td>
<td>0.4293 (25.1%)</td>
<td></td>
</tr>
</tbody>
</table>

Average: 0.3376 0.3781 +9.6% 0.3614 +7.4% 0.3976 +17.7%

Removing stop-words

- Function words: of, to, the, ...
  - have little meaning, no discriminating power
  - high frequency \( \Rightarrow \) large proportion of the index
  - 20% of the
- Treat as stop-words, remove from index
  - smaller index, faster retrieval, less noise
- Domain-specific (‘ chloride’, to be or not to be)
- Best policy: do not remove
  - make decisions at query/matching time
  - decrease weight during matching
  - what if a word occurs in every document?

Spelling variation

- 10-15% of web queries have spelling errors
  - mostly typos, phonetic errors
  - suggest corrections for any word not in spelling dictionary
- How to find suggestions?
  - similarity between user’s words and words in the dictionary
- Edit distance:
  - number of operations to transform user’s word into vocabulary word

Query-based stemming

- Big idea: do not stem words at indexing time
  - index the original word forms with no changes
  - at query time, expand word into stem-class
  - "political" \( \Rightarrow \) ("politician", "politics", "policy", "politician", ...)
- How to construct stem-classes?
  1. group words by n-character prefix
    - ("politician", "politics", "policy", "politician", ...)
  2. analyze co-occurrence of words within group
    - ("political") occurs in the same documents as ("policy")
    - ("politics") occurs in very different documents
- Very effective, but takes effort to get it right

Edit distance

- Damerau-Levenshtein distance:
  - minimum number of insertions + deletions + substitutions + transpositions of single characters
  - distance = 1:
    - extension \( \Rightarrow \) insert Explanation
    - pointer \( \Rightarrow \) pointer (deletion error)
    - marxmillow \( \Rightarrow \) marshallow (substitution error)
    - birmingham \( \Rightarrow \) birmingham (transposition error)
  - distance = 2:
    - directory \( \Rightarrow \) directory (deletion error)
    - generalized of Hamming distance
    - substitution only, same-length strings only
- O(mn) algorithm (dynamic programming)

Edit distance: dynamic programming

- Convert string A to string B ("jane" \( \Rightarrow \) "joe")
- \( d[i,j] \) \( \Rightarrow \) cost of converting \( A[:i] \) into \( B[:j] \)

\[
\begin{array}{c|cccc}
\text{i} & 0 & 1 & 2 & 3 \\
\hline
0 & 0 & & & \\
1 & & 1 & & \\
2 & & & & \\
3 & & & & \\
\end{array}
\]

Phonetic fingerprinting: Soundex

- find dictionary entry with smallest edit distance
  - compare word to every entry
  - O(V) complexity \( \Rightarrow \) expensive
- Speedups: compare only if
  - words begin with the same letter
  - words have similar length
  - words sound similar
- Phonetic fingerprinting:
  - words sound similar \( \Rightarrow \) same fingerprint
  - use fingerprint as key into hashtable
  - look-up correct spelling in O(1) time

Soundex

1. Keep the first letter (in upper case).
2. Replace these letters with hyphens: a.e.i.o.u,x,y,z.
3. Replace the other letters by numbers as follows:

<table>
<thead>
<tr>
<th>Letters</th>
<th>Numerals</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>5</td>
</tr>
<tr>
<td>i</td>
<td>5</td>
</tr>
<tr>
<td>j,k,q,x,z</td>
<td>9</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
</tr>
<tr>
<td>s</td>
<td>6</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
</tr>
<tr>
<td>w</td>
<td>6</td>
</tr>
<tr>
<td>y</td>
<td>6</td>
</tr>
</tbody>
</table>

4. Delete adjacent repeats of a number.
5. Delete the hyphens.
6. Keep the first three numbers or pad out with zeros.
Context-sensitive spelling

- "tink" → ("tank", "think") equally likely
  - "fish tank" → "fish tank" is much more probable
- P(w|w_j) ... language (context) model
  - which word the user is likely to type next
  - P(w) ... channel (noise) model
  - how the "channel" is likely to garble them
  - edit distance or estimates based on common types

Synonymy and Polysemy

- Natural language is ambiguous
  - synonymy: multiple words/expressions have the same meaning
    - car / automobile / vehicle, "banking crisis" vs. "financial meltdown"
  - polysemy: same word may have different meanings
    - bank: (river, money, memory), tank: (fish, military, gasoline)
  - Leads to mismatch of vocabulary in document/query
    - user may be unfamiliar with the domain (e.g. medical)
    - may not have full command of the language (non-native)
    - may have partial command of the language (partial)
  - extreme case: user does not speak language (ULR)
  - exacerbated by tendency to enter short queries
  - hard to disambiguate what the user meant without context

Dealing with linguistic ambiguity

- Goal: pick the right set of synonyms
- Possible approaches:
  - thesaurus / WordNet
  - statistically-related terms
  - Relevance Feedback
  - Pseudo-Relevance Feedback
  - Dimensionality Reduction
  - Latent Semantic Indexing (LSI)
  - Topic Models / Latent Dirichlet Allocation (LDA)

Synonyms: MeSH thesaurus

<table>
<thead>
<tr>
<th>MeSH Heading</th>
<th>Neck Pain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree Number</td>
<td>C01.597.003.768</td>
</tr>
<tr>
<td>Entry Term</td>
<td>Cervical Pain</td>
</tr>
<tr>
<td>Entry Term</td>
<td>Scoliosis</td>
</tr>
<tr>
<td>Entry Term</td>
<td>Neuralgia</td>
</tr>
<tr>
<td>Tree Number</td>
<td>C01.597.003.768</td>
</tr>
<tr>
<td>Entry Term</td>
<td>Anterior Neck Pain</td>
</tr>
<tr>
<td>Entry Term</td>
<td>Posterior Neck Pain</td>
</tr>
<tr>
<td>Entry Term</td>
<td>Cervicalgia</td>
</tr>
</tbody>
</table>

- excellent results in exact domains (medical, legal)
- errors in unrestricted search (lack context)
- development: time- and resource-intensive

WordNet

- Possible approaches:
  - 1. match synonyms
  - 2. similarity = distance
    - word−decompose−动物园
  - Issues to consider:
    - how do I phrase it?
    - are any of these synonyms?

Problems:
- no notion of context
- out-of-vocabulary meanings
  - liquid -> multiword? fluid? proxy server?

Example: statistical synonyms

<table>
<thead>
<tr>
<th>MIEH</th>
<th>EMEH</th>
<th>α²</th>
<th>Dice's coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>eukaryote</td>
<td>eukaryote</td>
<td>0.8</td>
<td>0.6</td>
</tr>
<tr>
<td>water</td>
<td>water</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>species</td>
<td>species</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>habitable</td>
<td>habitable</td>
<td>0.8</td>
<td>0.6</td>
</tr>
<tr>
<td>fish</td>
<td>fish</td>
<td>0.8</td>
<td>0.6</td>
</tr>
<tr>
<td>water</td>
<td>water</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>species</td>
<td>species</td>
<td>1.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

What is the largest possible value of MIEH?

- H_{α,B}
  - What pair of words will maximize MIEH?

Example: statistical synonyms (2)

- EMIEH, etc: based on binary co-occurrence
  - ignores frequencies, document length, clumping

Correlation coefficient:
- represent each word as a vector of occurrences:
  - "Angry" → 3.2 0.8 0.0 0.0 1.5
  - "Birds" → 5.4 0.0 0.0 4.1 0.0

Generalized Vector-Space Model

- Basic VSM: \( \text{sim}(Q, D) = \sum D_i Q_j \)
- Generalized vector-space: \( \text{sim}(Q, D) = \sum D_i Q_j S_{ij} \)

Example:
- Q = "pet lion"
  - D = "cats and dogs"
  - basic: \( \text{sim}(Q, D) = 0 \)
  - generalized: \( \text{sim}(Q, D) = Q_{lion} D_{cat} + Q_{pet} D_{cat} \)

Important issues:
- how many synonyms per term?
- how to set the weights \( S_{ij} \)
Relevance Feedback

- User labels ➔ new query:
  - should match more relevant documents
  - should match fewer non-relevant documents

Rocchio Feedback

- starting with qr, user marks |R| relevant, |NR| non-relevant docs
- define new query vector q_j as:
  \[ q_j = q_i + \alpha \left( \frac{1}{|R|} \sum_{j \in R} d_j - \frac{1}{|NR|} \sum_{j \in NR} d_j \right) \]
  - move query towards average relevant vector
  - move away from average non-relevant vector
  - \( \alpha \) ➔ parameters controlling relative importance of |R|, |NR|
  - typically \( \alpha = 0.5 \), tuned to optimize performance

Could we do the same without user involvement?

- top-ranked documents are usually relevant (on average)
- assume they are pseudo-relevant, use Rocchio with \( \beta = 0 \)

Can think of this as massive query expansion:

- add thousands of words from top-ranked docs to query
- statistical synonyms: words that co-occur with query
- handle synonymy in a domain-specific way (based on collection)
- note: co-occurrence with entire query, not individual words

- “fish tank” will not be expanded with tank-related minority terms

In practice ➔ a bit more complex to prevent query drift:
- focus of the query may shift, esp. for repeated expansion

Review: vector-space model

- Represent everything as a vector
  - documents, queries, user profiles, ...
  - dimensions ➔ words

- Term weighting: how important is word in a document
  - tf: how frequent is the term in D
  - normalization: is term unusual for D
  - idf: is it a content term or “linguistic glue”

- Invented by Karen Spärck Jones (1972)

- Vector similarity: is D “near” Q?
  - degree of match between document, query
  - project to sphere ➔ inner product \( \sum_a q_i \cdot d_i \)

Example: Rocchio Feedback

- vocabulary: \{run, lion, cat, dog, program\}
- original query: \( q_0 = [1,0,1,0,0] \) ➔ [1 run, 1 cat]
- relevant document: \( q_1 = [2,2,1,0,0] \) ➔ [2 run, 2 lion, 1 cat]
- non-relevant doc: \( q_2 = [2,0,1,0,3] \) ➔ [2 run, 1 cat, 3 program]

- \( q_j = q_0 + \alpha \left( \frac{\sum_{i \in R} q_i}{|R|} - \frac{\sum_{i \in NR} q_i}{|NR|} \right) \)
- \( \alpha = 1.0, \beta = 0.5 \)
- \( q_1 = q_0 + 0.5 \cdot \left( \frac{q_1}{|R|} - \frac{q_2}{|NR|} \right) \)

- \( q_1 = [1,0,1,0,0] + 0.5 \cdot [2,2,1,0,0] - 0.5 \cdot [2,0,1,0,3] \)
- \( = [2,2.5,1.5,0.5] \)
- \( = [2 \text{ run}, 2 \text{ lion}, 1.5 \text{ cat}] \)

Pseudo Relevance Feedback

- Run the original query, rank the documents
- User annotates 3 relevant, 1 non-relevant doc
- Construct new query representation
- Run it, compare rankings

PRF intuition: signal + noise

- Algorithm
  1. run original query
  2. take \( a_i \) top-ranked documents
  3. pick \( a_i \) highly-weighted terms
  4. add picked terms to the query
  5. run expanded query

- Why does it work? What if top-ranked non-relevant?
  - regularization: smoothness of scores over document space
  - redundancy: signal repeats, noise cancels out
  - one way to be relevant
  - relevant docs all similar
  - relevant words repeated
  - many ways to be non-relevant
  - noise words always different
Example: Relevance Models
(pseudo-relevance feedback based on generative models)

<table>
<thead>
<tr>
<th>president</th>
<th>lincoln</th>
<th>abraham lincoln</th>
<th>fishing</th>
<th>tropical fish</th>
</tr>
</thead>
<tbody>
<tr>
<td>bedroom</td>
<td>room</td>
<td>house</td>
<td>guest</td>
<td>america</td>
</tr>
<tr>
<td>white</td>
<td>american</td>
<td>new</td>
<td>caught</td>
<td>water</td>
</tr>
<tr>
<td>caught</td>
<td>aquatic</td>
<td>fair</td>
<td>christopher</td>
<td>tag</td>
</tr>
<tr>
<td>time</td>
<td>day</td>
<td>eat</td>
<td>sources</td>
<td>china</td>
</tr>
<tr>
<td>city</td>
<td>roof</td>
<td>tank</td>
<td>city</td>
<td>roof</td>
</tr>
<tr>
<td>people</td>
<td>animal</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fishermen</td>
<td>turpin</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boat</td>
<td>fishery</td>
<td>11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indexes semantic concepts

- Words not a good basis for a VS
- not independent: share meaning
- similarities are additive: T_i \cdot Q \cdot D_j
- Q, D: latent vectors
- \|Q\|: similarity (cosine similarity)

Can we index the meaning?

- dimensions = semantic concepts
  - fundamental, universal properties of things
  - large = hot, fast = cold, round = square
- words = combinations of concepts
  - "rain" = large, hot, round, ...
- documents = combinations of words

Latent Semantic Indexing (LSI)

- Goal: discover latent "semantic" dimensions from our data
- Approach: dimensionality reduction
  - singular value decomposition: C = T \cdot S \cdot D
  - \( T \): terms \times documents
  - \( S \): singular value (eigenvalues)
  - \( D \): rows = eigenvectors of \( C \)
  - take \( m = 50...500 \) eigenvectors with highest singular values
- \( T \) = "semantic" dimensions (m "clusters" of co-occurring terms)
- \( D \) = coordinates of docs in the (n-dimensional) "semantic" space
- \( \hat{Q} = T^T \cdot C \) = query as the "semantic" space
- "Dense" by similarity (cosine between \( \hat{Q} \) and the columns of \( D \))
- "Dense" comparisons = "magical" matches
- works for retrieving broad categories, bad for focused queries (why?)

LSI dimensions as "topics"
- Eigenvectors can be interpreted as "topics"

Vocabulary mismatch

- Punctuation: both split and merge, positional
- No word boundaries: character n-grams
- Morphology: Porter/Krovetz, character n-grams
- Spelling: edit distance, Soundex, contextual
- Synonymy and polysemy: (e.g. MeSH)
  1. Relevance feedback and pseudo-feedback
  2. In-domain thesaurus
  3. Statistical synonyms
  4. Topic models / semantic indexing