Dictionaries and tolerant retrieval

CE-324 : Modern Information Retrieval

Sharif University of Technology

M. Soleymani

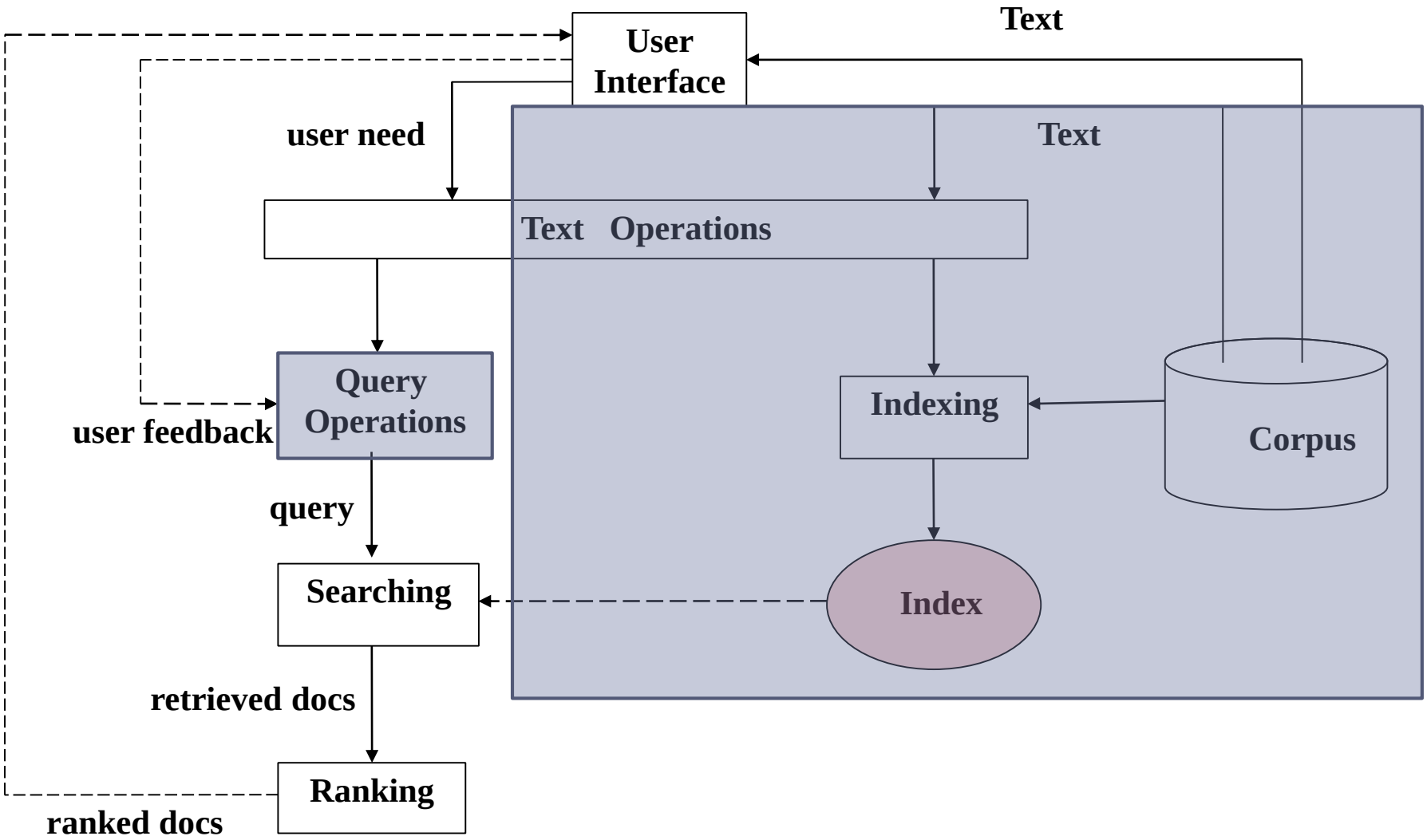
Fall 2015

Most slides have been adapted from: Profs. Nayak & Raghavan (CS-276, Stanford)

Topics

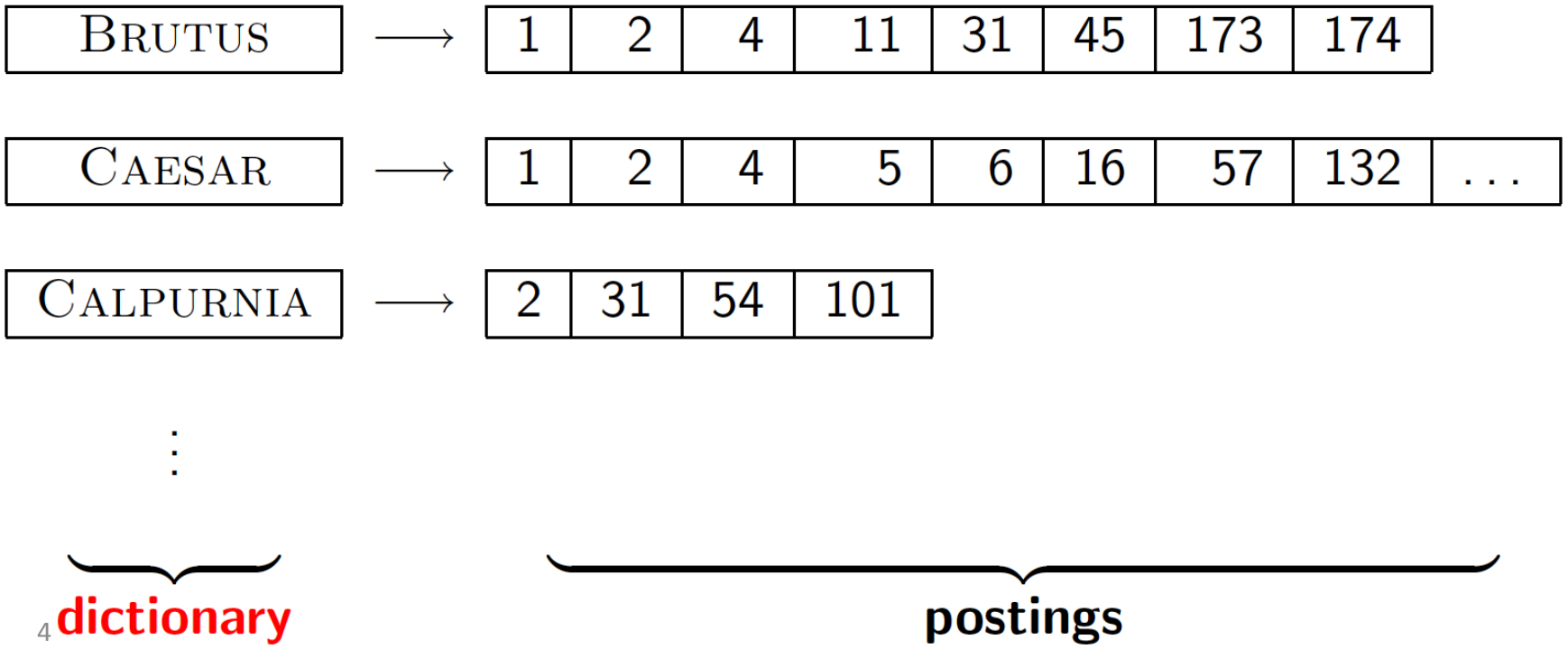
- "Tolerant" retrieval
  - Wild-card queries
  - Spelling correction
  - Soundex

Typical IR system architecture



Dictionary data structures for inverted indexes

- The dictionary data structure stores the term vocabulary, document frequency, pointers to each postings list … in what data structure?



Dictionary data structures

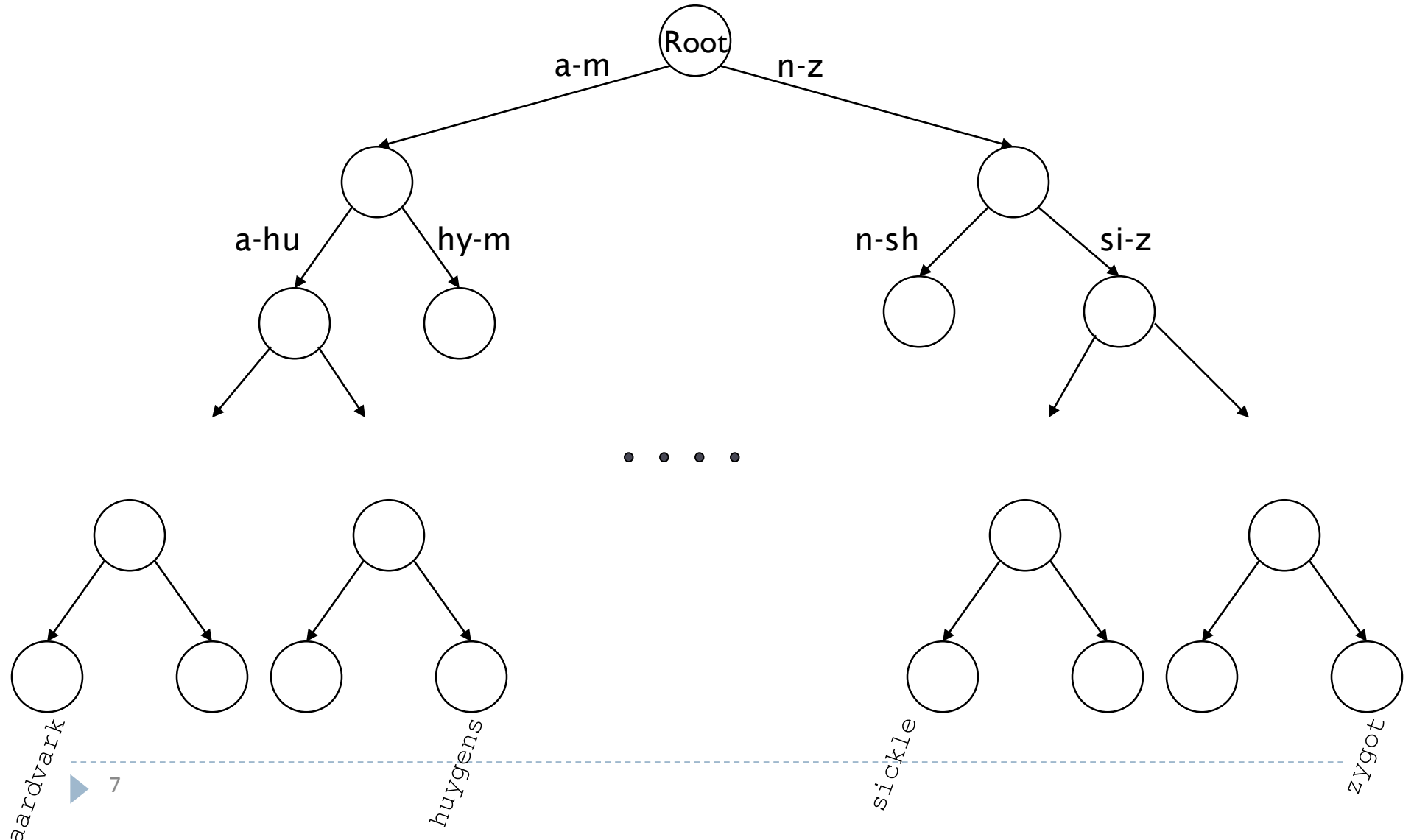
- Two main choices:
  - Hashtables
  - Trees
- Some IR systems use hashtables, some trees

Hashtables

- Each vocabulary term is hashed to an integer
  - (We assume you've seen hashtables before)

- Pros:
  - Lookup is faster than for a tree: O(1)
- Cons:
  - No easy way to find minor variants:
    - judgment/judgement
  - No prefix search [tolerant retrieval]
  - If vocabulary keeps growing, need to occasionally do the expensive operation of rehashing *everything*

Tree: binary tree



Trees

- Simplest: binary tree
- More usual: B-trees
- Trees require a standard ordering of characters and hence strings … but we typically have one

- Pros:
  - Solves the prefix problem (terms starting with *hyp*)
- Cons:
  - Slower: O(log *M*) [and this requires *balanced* tree]
  - Rebalancing binary trees is expensive
    - But B-trees mitigate the rebalancing problem

Wild-card queries: *

- Query: ***mon****
  - Any word beginning with "mon".
  - Easy with binary tree (or B-tree) lexicon: retrieve all words in range: ***mon*** ≤ ***w*** < ***moo***

- Query: ****mon***
  - Find words ending in "mon" (harder)
  - Maintain an additional tree for terms *backwards*. Can retrieve all words in range: ***nom*** ≤ ***w*** < ***non***.

- How can we enumerate all terms matching ***pro*cent*** ?

B-trees handle *'s at the end of a term

- How can we handle *'s in the **middle** of query term?

  co*tion

  co** AND ***tion

- Look up in the regular tree (for finding terms with the specified prefix) and the reverse tree (for finding terms with the specified suffix) and intersect these sets
  - Expensive
- Solutions:
  - permuterm index
  - k-gram index

Permuterm index

- For term ***hello***, index under:
  - ***hello$, ello$h, llohe, lohel, o$hell***

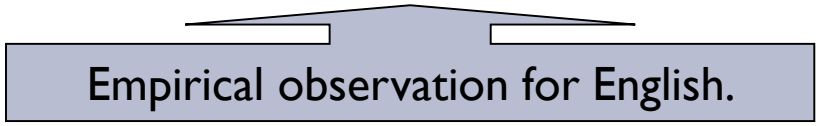
    where $ is a special symbol.

- Transform wild-card queries so that the *'s occur at the end

- Query: ***m*n***
  - ***m*n*** → ***n$m****
  - Lookup ***n$m**** in the permutation index
  - Lookup the matched terms in the standard inverted index

Permuterm query processing

- *Permuterm **problem**: ≈ quadruples lexicon size*



Bigram (*k*-gram) indexes

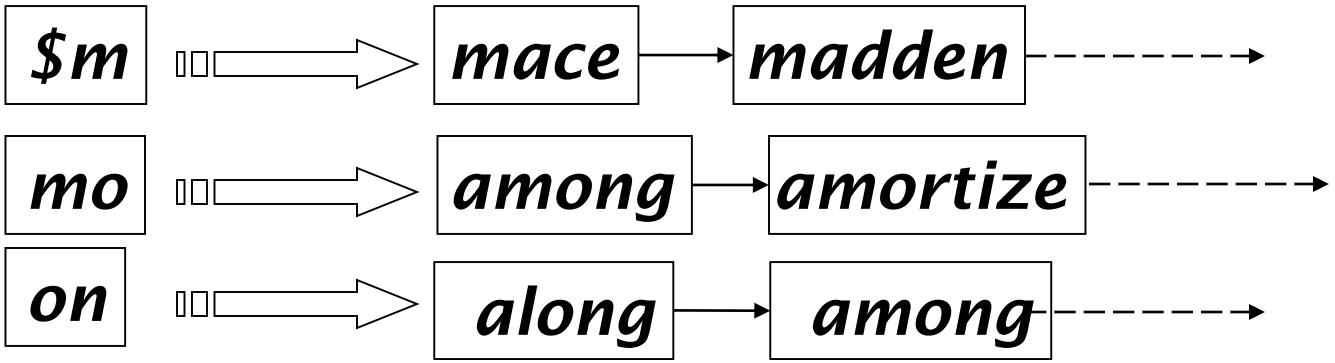
- Enumerate all *k*-grams (sequence of *k* chars)
  - e.g.,"***April is the cruelest month***" into 2-grams (*bigrams*)
    - $ is a special word boundary symbol

a, ap, pr, ri, il, l, i, is, s, t, th, he, e, $c, cr, ru, ue, el, le, es, st, t$, m, mo, on, nt, h

- Maintain a <u>*second*</u> inverted index
  - <u>*from bigrams to dictionary terms*</u> that match each bigram.

Bigram index example

- ***k*-gram index**: finds *terms* based on a query consisting of *k*-grams (here *k*=2).



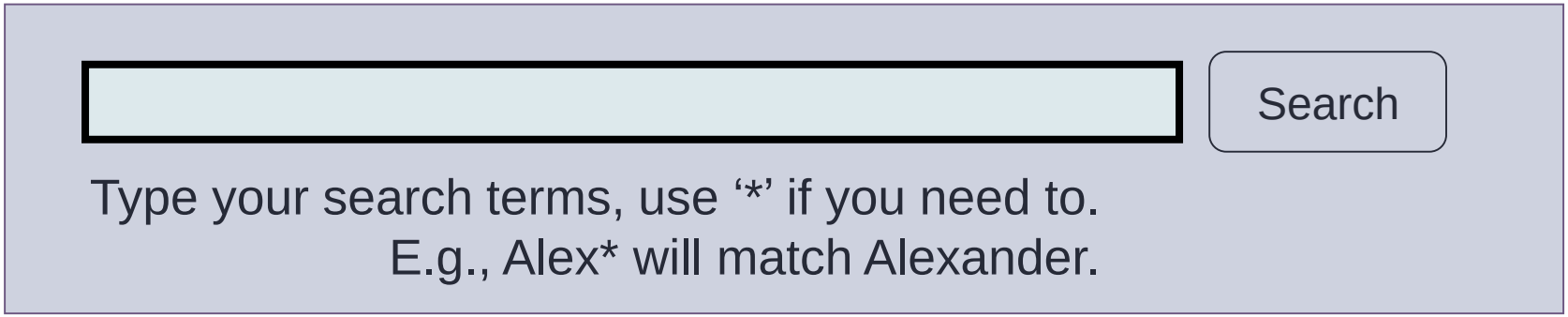
Bigram index: Processing wild-cards

- Query: ***mon****
  - → ***$m*** *AND* ***mo*** *AND* ***on***
  - But we'd enumerate ***moon*** (false positive).
- Must post-filter these terms against query.
- Run surviving ones through term-document inverted index.
- Fast, space efficient (compared to permuterm).

Processing wild-card queries

- Wild-cards can result in expensive query execution
  - pyth* AND prog*
  - As before, a Boolean query for each enumerated, filtered term (conjunction of disjunctions).

- If you encourage "laziness" people will respond!



Spelling correction

Google exprience

Search

About 2,420,000 results (0.26 seconds)

Web
Images
Maps
Videos
News
Shopping
More

Show search tools

Did you mean: ***experience***

How do you spell exprience
www.how-do-you-spell.com/**exprience**
What is the correct spelling of **exprience**. How do you spell **exprience**.

Urban Dictionary: exprience
www.urbandictionary.com/define.php?term=**exprience**
The **exprience** of starting an acid trip. **...** A movement that believes that women should have more sexual **expriences** that men and use the media to get there **...**

Integrity - Exprience The Difference | Facebook
www.facebook.com/pages/Integrity-**Exprience**-The.../161628437829
Integrity - **Exprience** The Difference, Tel Aviv-Yafo, Israel. 57 likes · 0 talking about this.

Share Your Exprience - Pediatric Oncall
www.pediatriconcall.com › Parent Corner
Article:- Hyderabad, which was slow to catch up with other major metros of India in surrogacy, has of late seen a change. Today, many fertility clinics in the City **...**

HoPE, Hands on Practical Exprience
www.indigoct.com/hope/
HoPE hands on practical **exprience ...** Years of **Exprience** number required. number b/n 1 and 100 required. number b/n 1 and 100 required. PHP Level **...**

customer exprience | Forrester Blogs
blogs.forrester.com/category/customer_**exprience**
2 days ago – Adele Sage. Many of the conversations I have with clients about voice of the customer programs center on ways the programs can improve and **...**

Spell correction

- Two principal uses
  - Correcting doc(s) being indexed
  - Correcting user queries to retrieve "right" answers

- Two main flavors:
  - Isolated word:
    - Check each word on its own for misspelling. Will not catch typos resulting in correctly spelled words (e.g., ***from → form)***
  - Context-sensitive:
    - Look at surrounding words,
      - e.g., ***I flew form Heathrow to Narita.***

Document correction

- Especially needed for OCR'ed docs
  - Can use domain-specific knowledge
    - E.g., OCR can confuse O and D more often than it would confuse O and I (that are adjacent on the keyboard and so more likely interchanged in typing query).
- But also: web pages
- Goal: the dictionary contains fewer misspellings
- But often we don't change docs and instead fix query-doc mapping

Lexicon

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

- Two basic choices for this
  - A standard lexicon such as
    - Webster's English Dictionary
    - An "industry-specific" lexicon (hand-maintained)
  - The lexicon of the indexed corpus (Including mis-spellings)
    - E.g., all words on the web
    - All names, acronyms etc.

Basic principles for spelling correction

- From correct spellings for a misspelled query choose the "nearest" one.
- When two correctly spelled queries are tied, select the one that is more common.
  - Query: grnt
  - Correction: grunt? grant?

Query mis-spellings

- We can either
  - Retrieve docs indexed by the correct spelling of the query when the query term is not in the dictionary, OR
  - Retrieve docs indexed by the correct spelling only when the the original query returned fewer than a preset number of docs , OR
  - Return several suggested alternative queries with the correct spelling
    - *Did you mean … ?*

Isolated word correction

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

- We'll study several alternatives for closeness
  - Edit distance (Levenshtein distance)
    - Weighted edit distance
  - *n*-gram overlap

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 ***dof*** to ***dog*** is 1
  - From ***cat*** to ***act*** is 2 (Just 1 with transpose.)
  - from ***cat*** to ***dog*** is 3.

Edit distance

- Generally found by dynamic programming.

```
EDITDISTANCE(s1, s2)
 1  int m[i, j] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8       do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1
 9                        m[i − 1, j] + 1,
10                        m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

Weighted edit distance

- As above, but the weight of an operation depends on the character(s) involved
  - keyboard errors
    - Example: ***m*** more likely to be mis-typed 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 distances

- A way: Given query,
  - enumerate all character sequences within a preset edit distance
  - Intersect this set with the list of "correct" words
  - Show terms you found to user as suggestions

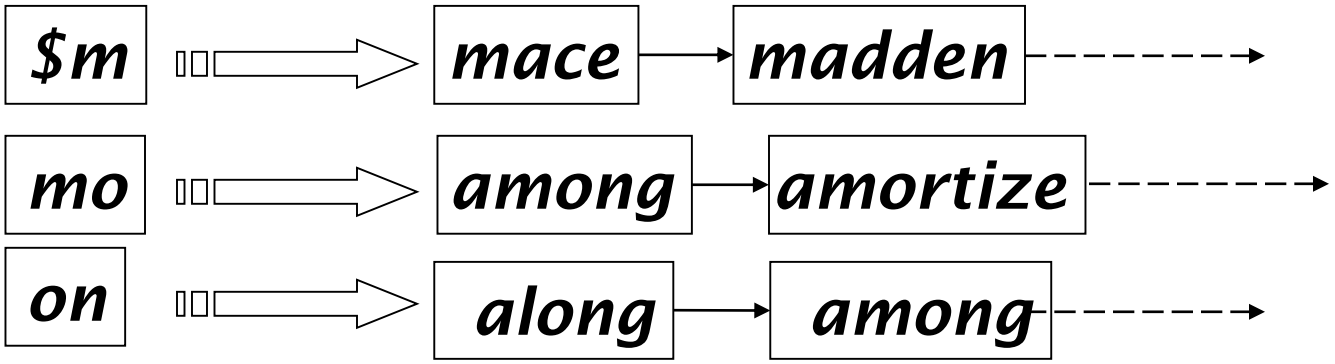
- Alternatively,
  - We can look up all possible corrections in our inverted index and return all docs … slow
  - We can run with a single most likely correction
    - disempower the user, but save a round of interaction with the user

Edit distance to all dictionary terms?

- Given a (mis-spelled) query – do we compute its edit distance to every dictionary term?
  - Expensive and slow
- How do we cut the set of candidate dictionary terms?
  - One possibility is to use *n*-gram overlap for this
    - This can also be used by itself for spelling correction.

n-gram overlap

- Enumerate all *n*-grams in the query
- Use the *n*-gram index for the lexicon to retrieve all terms matching an *n*-gram
- Threshold by number of matching *n*-grams
  - Variants – weights are also considered according to the keyboard layout, etc.



Example with trigrams

- Suppose the text is ***november***
  - Trigrams are *nov, ove, vem, emb, mbe, ber*.
- The query is ***december***
  - Trigrams are *dec, ece, cem, emb, mbe, ber*.
- So 3 trigrams overlap (of 6 in each term)

- How can we turn this into a normalized measure of overlap?

One option – Jaccard coefficient

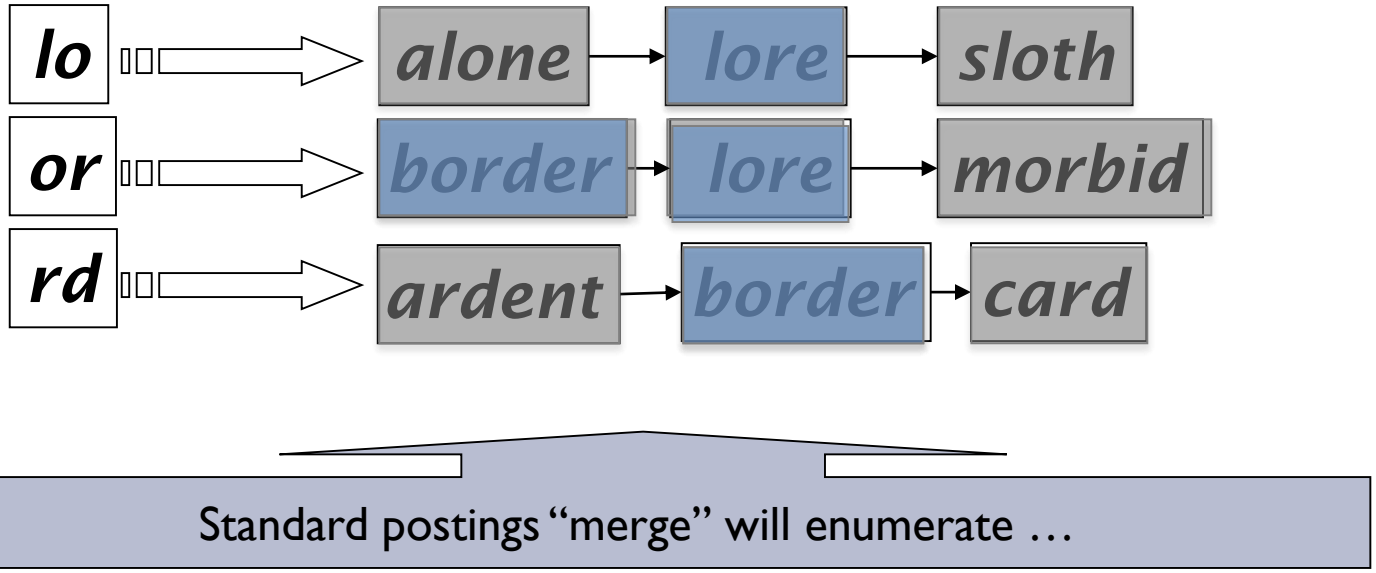
- A commonly-used measure of overlap between two sets:

$$JC(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

- Properties
  - X and Y don't have to be of the same size
  - Equals 1 when X and Y have the same elements and zero when they are disjoint
  - Always assigns a number between 0 and 1
    - Now threshold to decide if you have a match
    - E.g., if J.C. > 0.8, declare a match

Example

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



Adapt this example to using Jaccard measure.

Context-sensitive spell correction

- Phrase query: ***"flew form Heathrow"***
- Text: ***I flew from Heathrow to Narita.***
- We'd like to respond

  Did you mean "***flew from Heathrow***"?

  because no docs matched the query phrase.

Context-sensitive correction

- Need surrounding context to catch this.
- First idea: retrieve dictionary terms close to each query term
- Now try all possible resulting phrases with one word "fixed" at a time
  - ***flew from heathrow***
  - ***fled form heathrow***
  - ***flea form heathrow***
- **Hit-based spelling correction:** Suggest the alternative that has lots of hits.

Another approach

- Break phrase query into a conjunction of biwords
- Look for biwords that need only one term be corrected.
- Enumerate only phrases containing "common" biwords.

Soundex Algorithm

Soundex

- Class of heuristics to expand a query into phonetic equivalents
  - Language specific (mainly for names)
  - E.g., ***chebyshev*** → ***tchebycheff***

- Invented for the U.S. census … in 1918

Soundex – typical algorithm

- Turn every token to be indexed into a 4-character reduced form
- Do the same with query terms
- Soundex index: Build and search an index on the reduced forms
  - Used when the query calls for a soundex match

http://www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm#Top

Soundex algorithm

1. Retain the first letter of the word.
2. Change all occurrences of the following letters to '0':
   'A', E', 'I', 'O', 'U', 'H', 'W', 'Y' → 0
3. Change letters to digits as follows:
   - B, F, P, V → 1
   - C, G, J, K, Q, S, X, Z → 2
   - D, T → 3
   - L → 4
   - M, N → 5
   - R → 6

Soundex algorithm

4. Remove all pairs of consecutive digits.
5. Remove all zeros from the resulting string.
6. Pad the resulting string with trailing zeros and return the first four positions of the form:

   <uppercase letter> <digit> <digit> <digit>.

Example: ***Herman*** → H655.

Will ***hermann*** generate the same code?

Soundex

- Soundex is the classic algorithm
  - provided by most databases (Oracle, Microsoft, …)
- How useful is soundex?
  - Not very – for information retrieval
  - Okay for "high recall" tasks (e.g., Interpol), though biased to names of certain nationalities
- Zobel and Dart (1996) show that other algorithms for phonetic matching perform much better in the context of IR

What queries can we process?

- We have
  - Positional inverted index
  - Wild-card index
  - Spell-correction
  - Soundex

- Queries such as

(SPELL(moriset) /3 toron*to) OR SOUNDEX**(chaikofski)**

Resources

- IIR 3, MG 4.2
- Efficient spell retrieval:
  - K. Kukich. Techniques for automatically correcting words in text. ACM Computing Surveys 24(4), Dec 1992.
  - J. Zobel and P. Dart. Finding approximate matches in large lexicons. Software - practice and experience 25(3), March 1995. http://citeseer.ist.psu.edu/zobel95finding.html
  - Mikael Tillenius: Efficient Generation and Ranking of Spelling Error Corrections. Master's thesis at Sweden's Royal Institute of Technology. http://citeseer.ist.psu.edu/179155.html
- **Nice, easy reading on spell correction:**
  - Peter Norvig: How to write a spelling corrector

    http://norvig.com/spell-correct.html