Monsoon 2020

6 - Wild Card Queries

Information Retrieval

by

Dr. Rajendra Prasath

Indian Institute of Information Technology, Sri City, Chittoor

Sri City – 517 646, Andhra Pradesh, India

Recap: Phrase queries

- We want to be able to answer queries such as **“stanford university”** – as a phrase
- Thus the sentence “**I went to university at Stanford**” is not a match.
  - The concept of phrase queries has proven easily understood by users; one of the few “advanced search” ideas that works
  - Many more queries are *implicit phrase queries*
- For this, it no longer suffices to store only <*term* : *docs*> entries

WILD-CARD QUERIES

Wild-card queries: *

- ***mon*:*** find all docs containing any word beginning with "mon".
- Easy with binary tree (or B-tree) dictionary: retrieve all words in range: ***mon ≤ w < moo***
- ****mon:*** find words ending in "mon": harder
  - Maintain an additional B-tree for terms *backwards.*

Can retrieve all words in range: ***nom ≤ w < non.***

From this, how can we enumerate all terms meeting the wild-card query ***pro*cent*** ?

Query processing

- At this point, we have an enumeration of all terms in the dictionary that match the wild-card query.
- We still have to look up the postings for each enumerated term.
- E.g., consider the query:
- se*ate AND fil*er
- This may result in the execution of many Boolean AND queries.

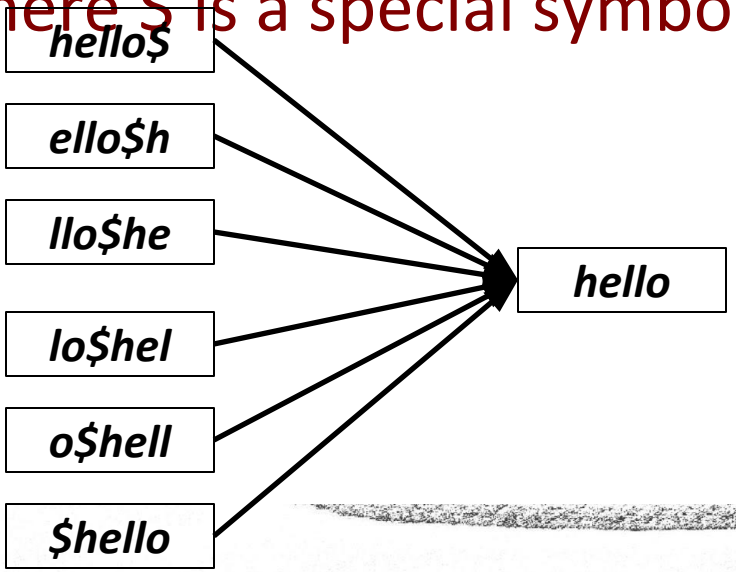
B-trees handle *'s at the end of a query term

- How can we handle *'s in the middle of query term?
  - co*tion
- We could look up co* AND *tion in a B-tree and intersect the two term sets
  - Expensive
- The solution: transform wild-card queries so that the *'s occur at the end
- This gives rise to the Permuterm Index.

Permuterm index

- Add a ***$*** to the end of each term
- Rotate the resulting term and index them in a B-tree
- For term ***hello***, index under:
  - ***hello$, ello$h, llohe, lohel, o$hell, $hello***

where $ is a special symbol.



Empirically, dictionary quadruples in size

Permuterm query processing

- (Add ***$***), rotate * to end, lookup in permuterm index
- Queries:
  - **X** lookup on **X$** ***hello$*** for ***hello***
  - **X*** lookup on $**X*** ***$hel**** for ***hel****
  - ***X** lookup on **X$*** ***llo$**** for ****llo***
  - ***X*** lookup on **X*** ***ell**** for ****ell****
  - **X*Y** lookup on **Y$X*** ***lo$h*** for ***h*lo***
  - **X*Y*Z** treat as a search for **X*Z** and post-filter

    For ***h*a*o***, search for ***h*o*** by looking up ***o$h**** and post-filter ***hello*** and retain ***halo***

Bigram (k-gram) indexes

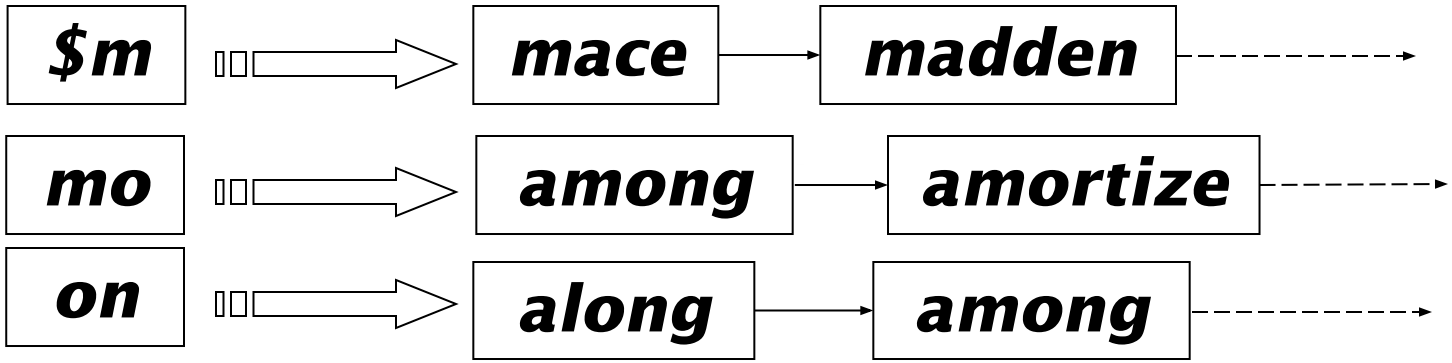
- Enumerate all *k*-grams (sequence of *k* chars) occurring in any term
- *e.g.,* from text "***April is the cruelest month***" we get the 2-grams (*bigrams*)

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

  - $ is a special word boundary symbol
- Maintain a *second* inverted index *from bigrams to dictionary terms* that match each bigram.

Bigram Index Example

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

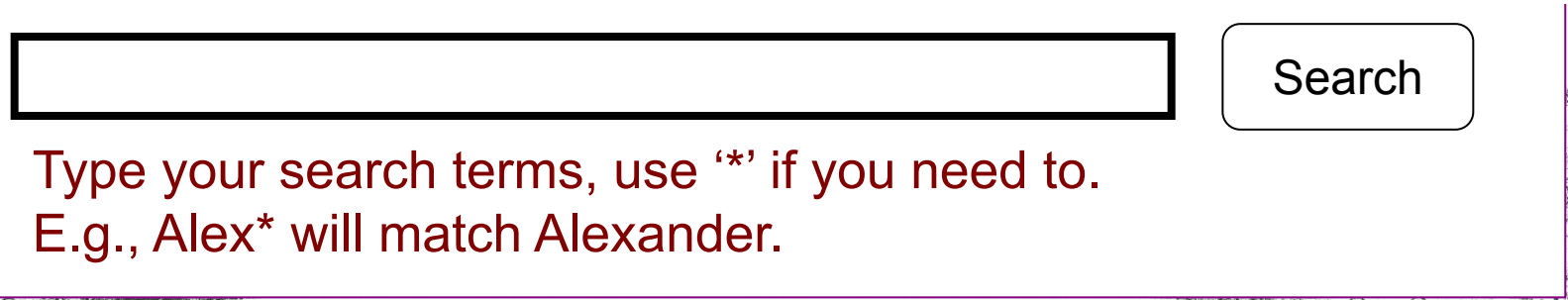


Processing wild-cards

- Query ***mon**** can now be run as
  - ***$m*** *AND* ***mo*** *AND* ***on***
- Gets terms that match AND version of our wildcard query.
- But we'd enumerate ***moon***.
- Must post-filter these terms against query.
- Surviving enumerated terms are then looked up in the term-document inverted index.
- Fast, space efficient (compared to permuterm).

Processing wild-card queries

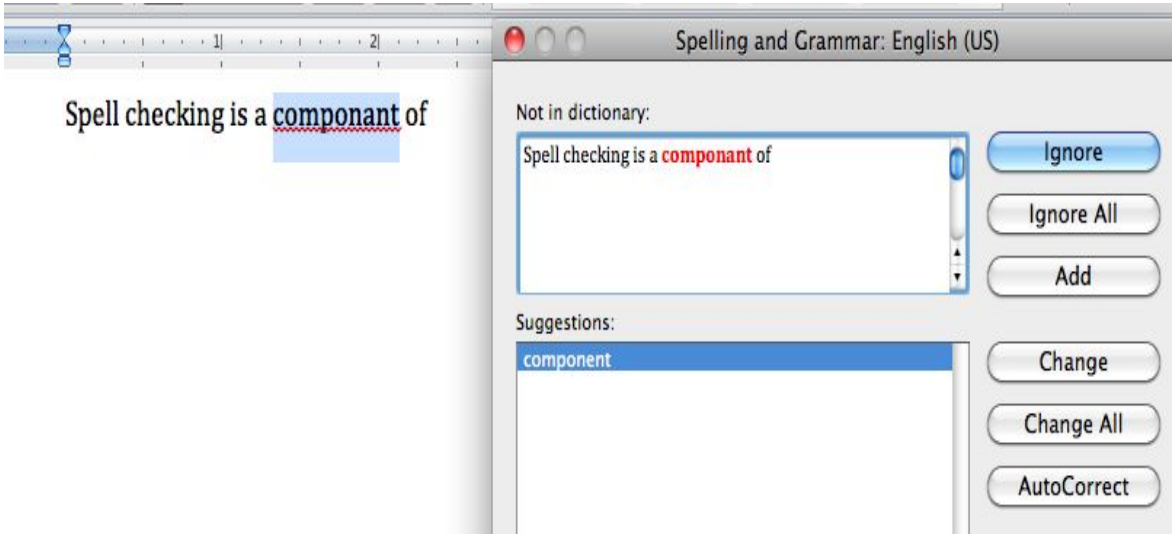
- As before, we must execute a Boolean query for each enumerated, filtered term.
- Wild-cards can result in expensive query execution (very large disjunctions…)
  - pyth* AND prog*
- If you encourage "laziness" people will respond!



SPELLING CORRECTION

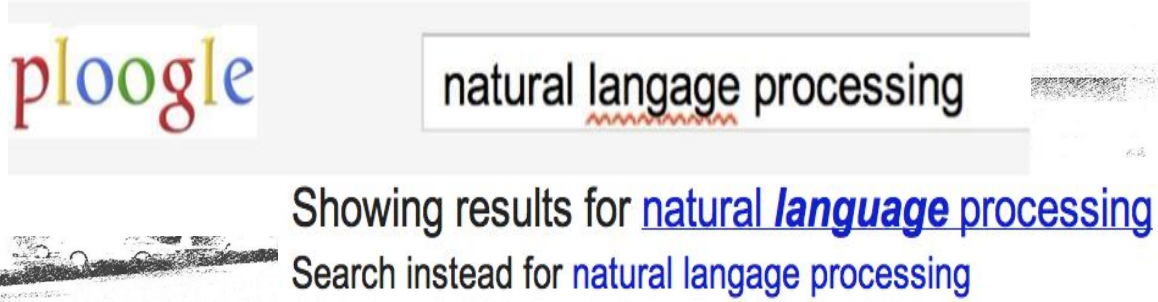
Apps For Spelling Correction

Word processing



Phones

Web search



Rates of spelling errors

Depends on the Appln: ~1–20% error rates

26%: Web queries Wang *et al.* 2003

13%: Retyping, no backspace: Whitelaw *et al.* English & German

7%: Words corrected retyping on phone-sized organizer

2%: Words uncorrected on organizer Soukoreff & MacKenzie 2003

1-2%: Retyping: Kane and Wobbrock 2007, Gruden et al. 1983

Spelling Tasks

- Spelling Error Detection
- Spelling Error Correction:
  - Autocorrect
    - hte □ the
  - Suggest a correction
  - Suggestion lists

Types of spelling errors

- Non-word Errors
  - graffe giraffe
- Real-word Errors
  - Typographical errors
    - three there
  - Cognitive Errors (homophones)
    - piece peace,
    - too two
    - your you're
- Non-word correction was mainly context insensitive
- Real-word correction almost needs to be context sensitive

Non-word spelling errors

- Non-word spelling error detection:
  - Any word not in a dictionary is an error
  - The larger the dictionary the better … up to a point
  - (The Web is full of mis-spellings, so the Web isn't necessarily a great dictionary …)
- Non-word spelling error correction:
  - Generate candidates: real words that are similar to error
  - Choose the one which is best:
    - Shortest weighted edit distance
    - Highest noisy channel probability

Real word & non-word spelling errors

- For each word *w*, generate candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spellings***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel view of spell errors
  - Context-sensitive – so have to consider whether the surrounding words "make sense"
  - *Flying form Heathrow to LAX Flying from Heathrow to LAX*

Terminology

- We just discussed *character bigrams and k-grams*:
  - *st, pr, an …*
- We can also have *word bigrams and n-grams*:
  - *palo alto, flying from, road repairs*

Summary

In this class, we focused on:

(a) Wild card Queries

(b) Permuterm index

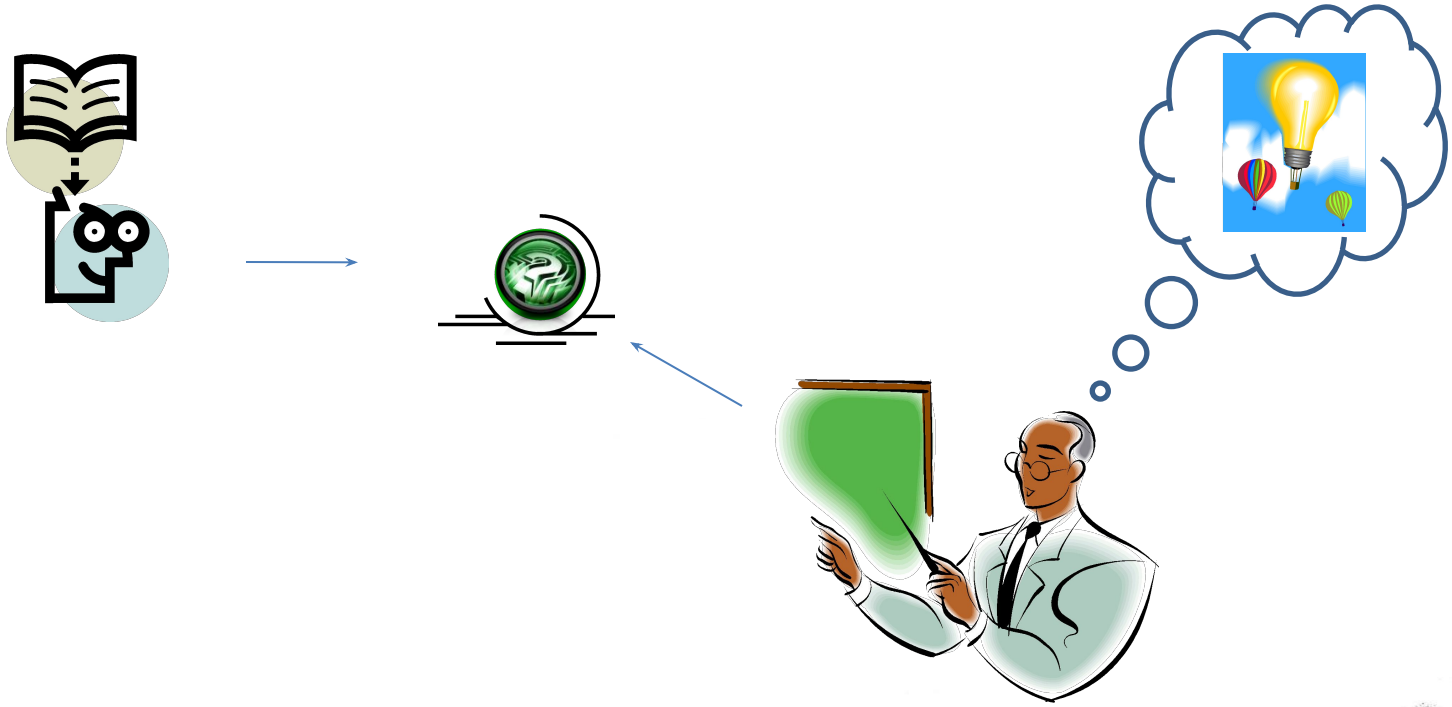
(c) Spelling Corrections

Acknowledgements

Thanks to ALL RESEARCHERS:

1. Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.
2. Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
3. Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
4. Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
5. Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
6. Prof. Mandar Mitra, Indian Statistical Institute, Kolkata (https://www.isical.ac.in/~mandar/)

Thanks …



… Questions ???