Dictionary building

Caroline Barrière
January 15th 2020

Outline

- What would an ideal IR system do?
- Dictionary building
    - Document characterization
    - Tokenization
    - Case folding
    - Normalization
    - Stemming / lemmatization
- Flexible retrieval
    - Wildcards

References

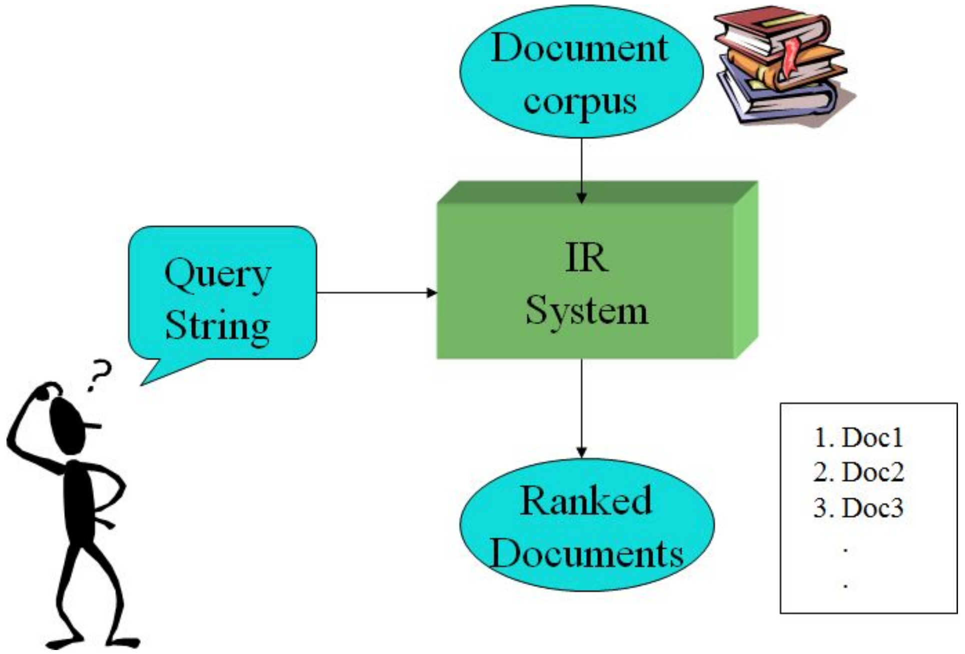
Textbook

Introduction to IR, by Manning et al. 2009 (IIR)
Chapter 2
https://nlp.stanford.edu/IR-book/pdf/02voc.pdf
Chapter 3, Sections 3.2
https://nlp.stanford.edu/IR-book/pdf/03dict.pdf

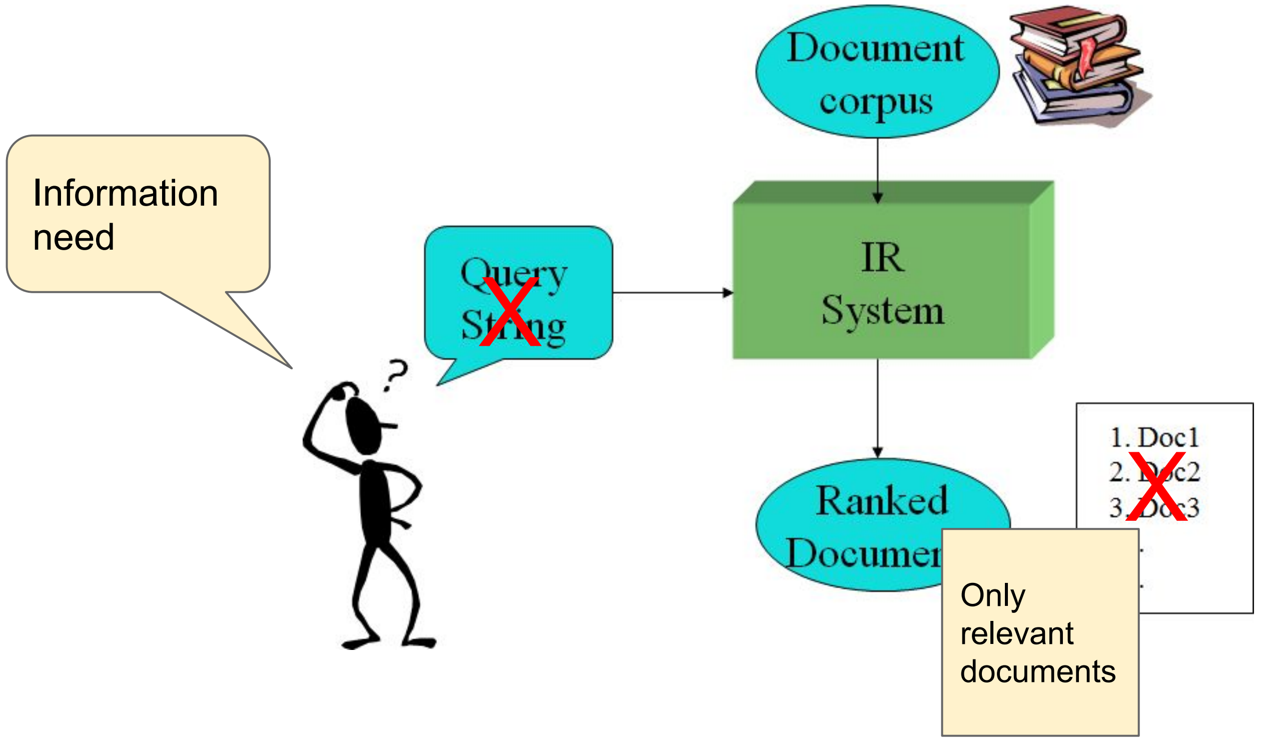
Slides
Stanford slides on Term Vocabulary
https://web.stanford.edu/class/cs276/handouts/lecture2-dictionary-handout-1-per.pdf

Ideal Information Retrieval

What we have


Document corpus
Query String
IR System
Ranked Documents
1. Doc1
2. Doc2
3. Doc3
.
.

What we want


Information need
Query String
Document corpus
IR System
Ranked Documents
1. Doc1
2. Doc2
3. Doc3
Only relevant documents

Let's look at these examples

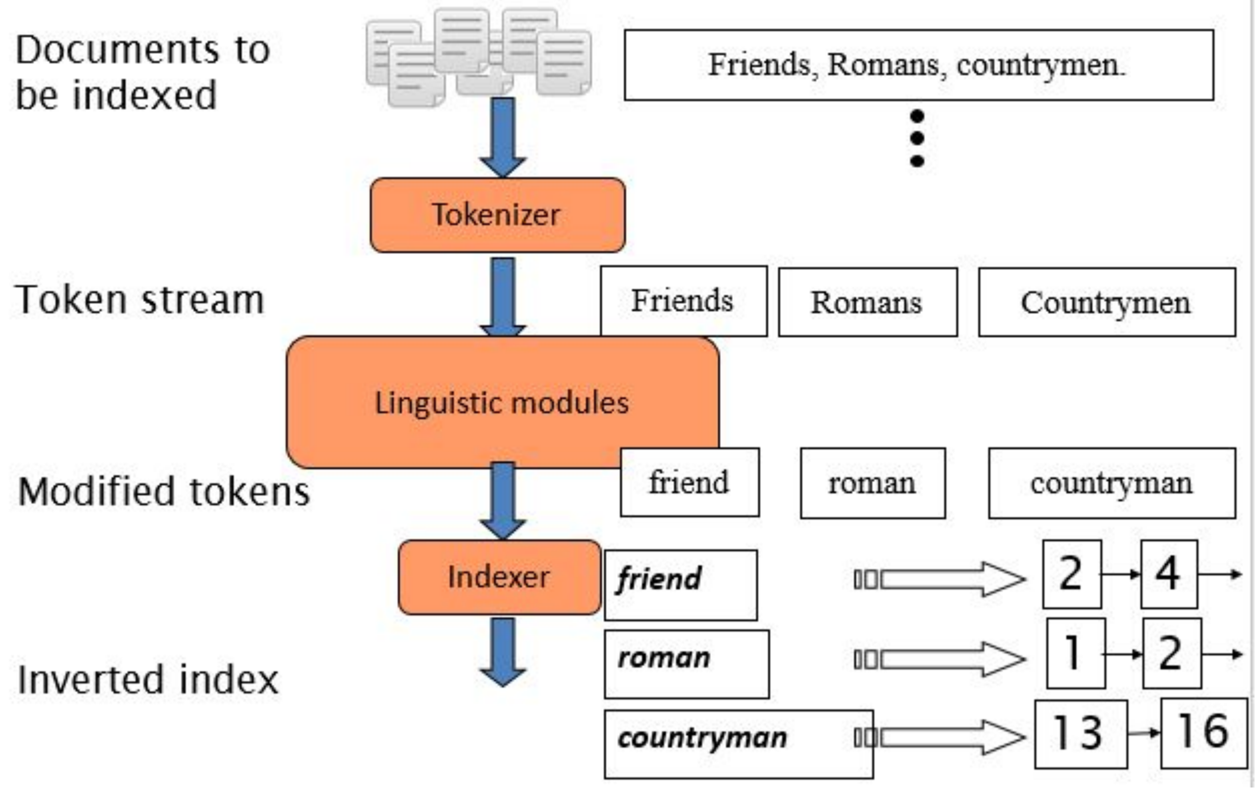
Email information need:

- Julie's (can't remember her last name) email about her dinner invitation this week-end.
- J. Bellinger's email about smoking regulations on campus.
- The email I sent to the director about going to the IR conference in San José.

Dictionary building

Dictionary building

Dictionary building process



Dictionary building

Document characterization

- What format is it in?
  - pdf/word/excel/html?
- What language is it in?
- What character set is in use?
  - (CP1252, UTF-8, …)

More info in Section 2.1 of textbook

Document characterization

Example semi-structured document

Vanilla System

https://catalogue.uottawa.ca/en/courses/csi/

```html
<p class="courseblocktitle noindent"><strong>CSI 2101 Discrete Structures (3
units)</strong></p>
<p class="courseblockdesc noindent">
Discrete structures as they apply to computer science, algorithm analysis and design.
Predicate logic. Review of proof techniques; application of induction to computing
problems. Graph theory applications in information technology. Program correctness,
preconditions, postconditions and invariants. Analysis of recursive programs using
recurrence relations. Properties of integers and basic cryptographical
applications.</p><p class="courseblockextra noindent"><strong>Course Component:
</strong>Discussion Group, Lecture</p><p class="courseblockextra highlight
noindent">Prerequisite: <a href="/search/?P=MAT%201348" title="MAT 1348"
class="bubblelink code" onclick="return showCourse(this, 'MAT 1348');">MAT
1348</a>.</p></div><div class="courseblock">
<p class="courseblocktitle noindent"><strong>CSI 2110 Data Structures and
Algorithms (3 units)</strong></p>
<p class="courseblockdesc noindent">
The concept of abstract data types. Simple methods of complexity analysis. Trees.
The search problem: balanced trees, binary-trees, hashing. Sorting. Graphs and
simple graph algorithms: traversal, minimum spanning tree. Strings and pattern
matching.</p><p class="courseblockextra noindent"><strong>Course Component:
</strong>Laboratory, Lecture, Tutorial</p><p class="courseblockextra highlight
noindent">Prerequisites: <a href="/search/?P=ITI%201121" title="ITI 1121"
class="bubblelink code" onclick="return showCourse(this, 'ITI 1121');">ITI 1121</a>,
<a href="/search/?P=MAT%201348" title="MAT 1348" class="bubblelink code"
onclick="return showCourse(this, 'MAT 1348');">MAT 1348</a>.</p></div><div
class="courseblock">
```

Document characterization

Example semi-structured document

Reuters collection

Final system

```
<REUTERS TOPICS="NO" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="5545" NEWID="2">
<DATE>26-FEB-1987 15:02:20.00</DATE>
<TOPICS></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<TITLE>STANDARD OIL &lt;SRD> TO FORM FINANCIAL UNIT</TITLE>
<DATELINE>    CLEVELAND, Feb 26 - </DATELINE>

<BODY>Standard Oil Co and BP North America Inc said they plan to form a venture
to manage the money market
borrowing and investment activities of both companies.BP North America is a
subsidiary of British Petroleum Co Plc &lt;BP>, which also owns a 55 pct interest in
Standard Oil. The venture will be called BP/Standard Financial Trading
and will be operated by Standard Oil under the oversight of a
joint management committee.
 Reuter &#3;</BODY></TEXT>
</REUTERS>
```

Text processing

Tokenization issues

- ***Finland's capital*** →

  Finland AND ***s***? ***Finlands***? ***Finland's***?
- ***Hewlett-Packard*** → ***Hewlett*** and ***Packard*** as two tokens?
  - ***state-of-the-art***: break up hyphenated sequence.
  - ***co-education***
  - ***lowercase***, ***lower-case***, ***lower case*** ?
  - It can be effective to get the user to put in possible hyphens
- ***San Francisco***: one token or two?
  - How do you decide it is one token?

Text processing

Tokenization numbers

- ***3/20/91*** ***Mar. 12, 1991*** ***20/3/91***
- ***55 B.C.***
- ***B-52***
- ***My PGP key is 324a3df234cb23e***
- ***(800) 234-2333***

- Older IR systems may not index numbers
  - But often very useful: think about things like looking up error codes/stacktraces on the web
- Will often index “meta-data” separately
  - Creation date, format, etc.

Text processing

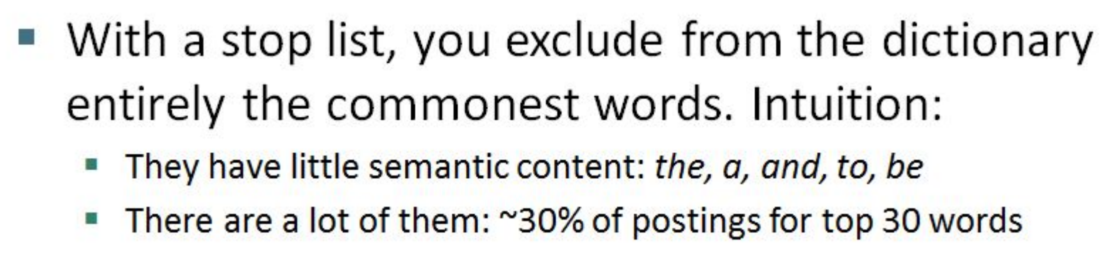
Tokenization

language specific issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***
      - Until at least 2003, it didn't on Google
        - Internationalization!
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German retrieval systems benefit greatly from a **compound splitter** module

Text processing

Stopwords



- With a stop list, you exclude from the dictionary entirely the commonest words. Intuition:
  - They have little semantic content: *the, a, and, to, be*
  - There are a lot of them: ~30% of postings for top 30 words

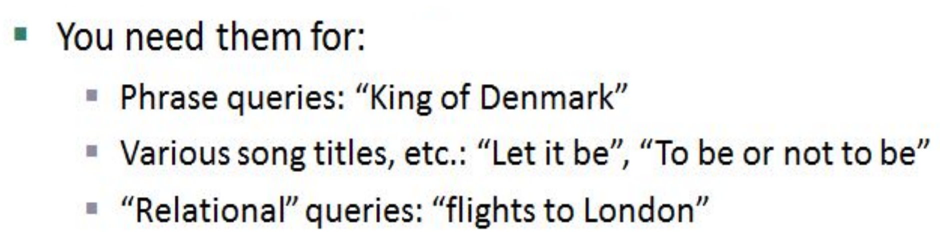
You can just search "stopword list":
https://www.ranks.nl/stopwords

Let's look at Google NGram Viewer
https://books.google.com/ngrams

Text processing

Stopwords

Should we really remove stopwords??
Efficiency / Recall trade-off



- You need them for:
  - Phrase queries: “King of Denmark”
  - Various song titles, etc.: “Let it be”, “To be or not to be”
  - “Relational” queries: “flights to London”

Text processing

Normalization

- We may need to “normalize” words in indexed text as well as query words into the same form
  - We want to match ***U.S.A.*** and ***USA***
- Result is terms: a term is a (normalized) word type, which is an entry in our IR system dictionary
- We most commonly implicitly define equivalence classes of terms by, e.g.,
  - deleting periods to form a term
    - ***U.S.A., USA***
  - deleting hyphens to form a term
    - ***anti-discriminatory, antidiscriminatory***

Text processing

Normalization

language specific issues

- Accents: e.g., French ***résumé*** vs. ***resume*.**
- Umlauts: e.g., German: ***Tuebingen*** vs. ***Tübingen***
  - Should be equivalent

Text processing

Normalization

- Tokenization and normalization may depend on the language and so is intertwined with language detection

- Crucial: Need to “normalize” indexed text as well as query terms identically

Text processing

Case folding

- Reduce all letters to lower case
  - exception: upper case in mid-sentence?
    - e.g., General Motors
    - Fed vs. fed
    - SAIL vs. sail
  - Often best to lower case everything, since users will use lowercase regardless of ‘correct’ capitalization…

Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* $\rightarrow$ *be*
  - *car, cars, car's, cars'* $\rightarrow$ *car*
- *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*
- Lemmatization implies doing “proper” reduction to dictionary headword form

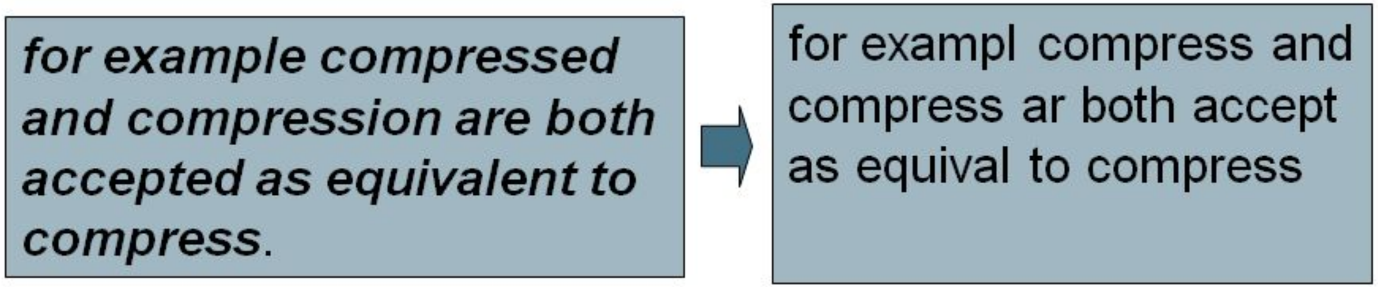
Text processing

Stemming

- Reduce terms to their “roots” before indexing
- “Stemming” suggests crude affix chopping
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

Text processing

Stemming example



for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Text processing

Stemming algorithm

Porter Stemmer

Most common stemmer for English.

Set of replacements performed in phases. Rule-based.
Word length dependent.

- *sses* $\rightarrow$ *ss*
- *ies* $\rightarrow$ *i*
- *ational* $\rightarrow$ *ate*
- *tional* $\rightarrow$ *tion*

Text processing

Stemming

Porter Stemmer code

Python NLTK package includes Porter Stemmer:
http://www.nltk.org/howto/stem.html

Java OpenNLP package includes Porter Stemmer:
https://opennlp.apache.org/docs/1.7.2/apidocs/opennlp-tools/opennlp/tools/stemmer/PorterStemmer.html

Or stand-alone java implementation
https://alvinalexander.com/java/jwarehouse/lucene-1.3-final/src/java/org/apache/lucene/analysis/PorterStemmer.java.shtml

Text processing

Stemming
How useful?

- English: very mixed results. Helps recall for some queries but harms precision on others
  - E.g., operative (dentistry) ⇒ oper
- Definitely useful for Spanish, German, Finnish, …
  - 30% performance gains for Finnish!

Text processing

Let's work through some examples

CSI 3131 Operating Systems (3 units)

Principles of operating systems. Operating systems design issues. Process management, process scheduling, concurrency issues. CPU scheduling. Memory management. Virtual memory. Mass storage systems. Input/Output system. File system. Security and protection. Examples of operating systems.

1. Tokenization
2. Case folding
3. Stopword removal
4. Normalization
5. Stemming versus lemmatization
6. Phrase indexing

Text processing

Let's work through some examples

CSI 4130 Computer Graphics (3 units)

Interactive computer graphics. Display data structures and procedures. Graphics pipeline. Geometric transformations. Viewing in three dimensions. Illumination and color models. Object modelling in 2D and 3D.

1. Tokenization
2. Case folding
3. Stopword removal
4. Normalization
5. Stemming versus lemmatization
6. Phrase indexing

Text processing

Let's work through some examples

CSI 4108 Cryptography (3 units)

The notion of secure communication. Building secure cryptosystems based on the assumption of computational hardness. Cryptographic one-way functions, trap-door functions, pseudorandom generators, and public/private-key encryption schemes. Computational indistinguishable and unpredictability. Digital signature and message authentication. Zero-knowledge/interactive proof systems. Application to e-commerce and e-trade.

1. Tokenization
2. Case folding
3. Stopword removal
4. Normalization
5. Stemming versus lemmatization
6. Phrase indexing

Flexible retrieval

Flexible retrieval

Wildcard queries

- ***mon*:*** find all docs containing any word beginning with “mon”.
- ****mon:*** find words ending in “mon”: harder

Flexible retrieval

Wildcard queries

within words

- E.g., consider the query:
  se*ate *AND* ***fil*er***

Flexible retrieval

Wildcard queries

Bigram 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

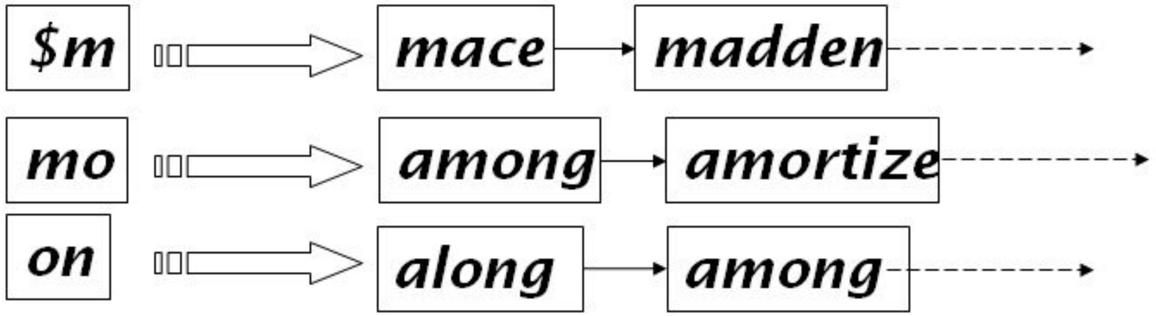
  - $ is a special word boundary symbol
- Maintain a <u>*second*</u> inverted index <u>*from bigrams to dictionary terms*</u> that match each bigram.

Flexible retrieval

Wildcard queries

Bigram indexes

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



Flexible retrieval

Wildcard queries

Bigram indexes

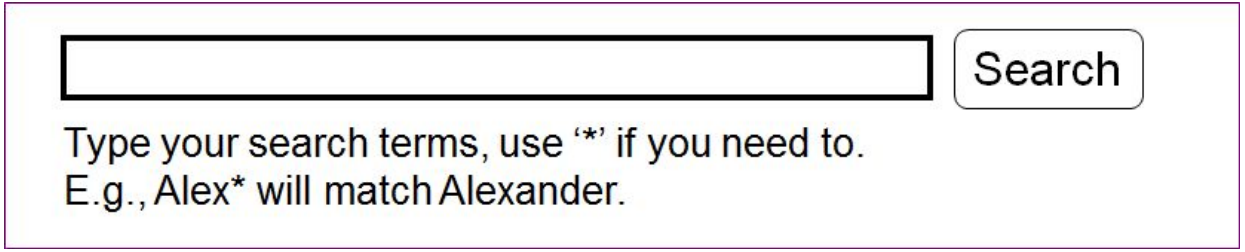
- Query ***mon**** can now be run as
  - ***$m** AND **mo** AND **on***
- Gets terms that match AND version of our wildcard query.

- Must post-filter these terms against query.
- Surviving enumerated terms are then looked up in the term-document inverted index.

Flexible retrieval

Wildcard queries

- Wild-cards can result in expensive query execution (very large disjunctions…)
  - pyth* AND prog*
- If you encourage "laziness" people will respond!



Type your search terms, use '*' if you need to.
E.g., Alex* will match Alexander.