# 情報検索システム特論

## Advanced Information Retrieval Systems

## 第6回 Lecture #6

2023-05-15

湯川 高志 Takashi Yukawa

# Query Languages

# Different Kinds of Queries

- Keyword-based querying
  - Single-word queries
  - Context queries
  - Boolean queries
  - Natural language
- Pattern matching
- Structural queries
  - Fixed structure
  - Hypertext
  - Hierarchical structure

# Keyword-Based Querying

- A query is composed of keywords
- The documents containing such keywords are searched for

- Keyword-based queries are popular
  - Intuitive
  - Easy to express
  - Allow for fast ranking

# Keyword-Based Querying (cont'd)

- A query can be
  - Simply a word
  - A more complex combination of operations involving several words

# Single-Word Queries

- The most elementary query
- Text documents are assumed to be essentially long sequences of words
- Definition of a “word”
  - The alphabet is split into “letters” and “separators”
  - A word is a sequence of letters surrounded by separators
- The result of word queries is the set of documents containing at least one of the words of the query
- The resulting documents are ranked according to a degree of similarity to the query

# Support for Ranking

- To support ranking, two common statistics on word occurrence inside texts are commonly used

- “term frequency”
  - counts the number of times a word appears inside a document
- “inverse document frequency”
  - counts the number of documents in which a word appears

# Context Queries

- To search words in a given ***context***, that is, near other words
  - Words which appear near each other may signal a higher likelihood of relevance
- Phrase
  - A sequence of single-word queries
- Proximity
  - A more relaxed version of the phrase query
  - A sequence of single words or phrases is given, together with a maximum allowed distance between them

# Boolean Queries

- The oldest form of combining keyword queries
- A Boolean query
  - A syntax composed of atoms (basic queries) that retrieve documents
  - Boolean operators which work on their operands and deliver sets of documents
- A query syntax tree is naturally defined

# Pattern Matching

- More specific query formulations which allow the retrieval of pieces of text that have some property
- These data retrieval queries are useful for
  - Linguistics
  - Text statistics
  - Data extraction

# “Pattern” in Pattern Matching

- A set of syntactic features that must occur in a text segment
  - Those segments satisfying the pattern specification are said to “match”

# Types of Patterns (1)

- Words
  - A string which must be a word in the text
  - The most basic pattern
- Prefixes
  - A string which must form the beginning of a text word
  - Ex: “comput” → the documents containing words such as “computer”, “computation”, “computing”, … are retrieved
- Suffixes
  - A string witch must form the termination of a text word
  - Ex” “ters” → the documents containing words such as “computers”, “testers”, “painters”, … are retrieved

# Types of Patterns (2)

- Substrings
  - A string which can appear within a text word
  - Ex: “tal” → the documents containing words such as “coastal”, “talk”, “metallic”, … are retrieved
- Ranges
  - A pair of string which match any word lying between them in lexicographical order
  - Ex: the range between words “held” and “hold” → retrieved string such as ”hoax”, “hissing”, …
- Allowing errors
  - A word together with an error threshold
  - Retrieves all text words witch are “similar” to the given word
  - “similar” = the pattern or the text may have errors (coming from typing, spelling …)
    - Levenshtein distance or simply edit distance

# Edit Distance

- The minimum number of
  - Character insertions
  - Character deletions
  - Character replacements
- needed to make them equal

- Example
  - “survey” : “surgery”
    - Replace “v” → “g”
    - Insert “r”
  - Edit distance = 2

# Types of Patterns (3)

- Regular expressions
  - A general pattern built by
    - Simple strings
    - The operators
      - Union
        - If e1 and e2 are regular expressions, then (e1|e2) matches what e1 or e2 matches
      - Concatenation
        - If e1 and e2 are regular expressions, the occurrence of (e1 e2) are formed by the occurrence of e1 immediately followed by those of e2
      - Repetition
        - If e is a regular expression, then (e*) matches a sequence of zero or more contiguous occurrences of e

# Details of Regular Expressions

# Quick Start of Regex

- Literal characters: match the occurrence of that character
- . : matches a single character
- ? : makes the preceding token in the regular expression optional
- + : to attempt to match the preceding token once or more
- *: to attempt to match the preceding token zero or more times
- | : equivalent of "or"
- ^: matches at the start of the string
- $: matches at the end of the string

(Anchors do not match any characters. They match a position)

- [ ]: matches only one out of several characters
  - : You can use a hyphen inside a character class to specify a range of characters
- ( ): create a capturing group

# Examples

- Attempting to match dates like “2003-04-25”
  - [0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]
  - [0-9]+-[0-9]+-[0-9]+
- Attempting to match Japanese e-mail address
  - [0-9a-zA-Z_-]+@([0-9a-zA-Z-]+¥.)+jp
- Attempting to match “Japan” or “Japanese”
  - Japan(ese)?
- Attempting to match “zooooooooooooooooooom” (two or more “o”)
  - zooo*m

# Special Characters

- Special characters (= metacharacters)
  - certain characters for special use
- Non-Special characters = Literal characters
    - a single literal character will match the first occurrence of that character in the string

# The Dot

- .
  - The dot matches (almost) any character
  - The only exception are newline characters

# Character Classes or Character Sets

- [ ]
  - tell the regex engine to match only one out of several characters

    [abcde]: matches “a”,”b”,”c”,”d”, or “e”
  - Typing a caret after the opening square bracket will negate the character class

    [^abcde]: matches any character not in the character class (a,b,c,d,e)
  - You can use a hyphen "–" inside a character class to specify a range of characters.

    [a–z]: matches a single character between “a” and “z”

    [0–9a–zA–Z]: a single digit, single lower case letter, or single upper case letter

# Quantifier

- *
  - attempting to match the preceding token zero or more times
    - "fo* " matches "fo", "foo", or "f"
- +
  - attempting to match the preceding token once or more
    - "fo+" matches "fo" or "foo", but "f"
- ?
  - attempting to match the preceding token zero times or once, in effect making it optional
    - "fo?" matches "f" or "fo"

# Quantifier (cont'd)

- Limiting repetition: The syntax is {min,max} , where min is a positive integer number indicating the minimum number of matches, and max is an integer equal to or greater than min indicating the maximum number of matches
- {n}
  - matches n times repetition
    - "fo{2}" matches "foo"
- {n,}
  - matches greater than or equal to n times repetition
    - "fo{2,}" matches "foo", "fooo", "foooo", or . . .
- {n,m}
  - matches greater than or equal to n times and less than or equal to m times repetition
    - "fo{2,3}" matches "foo" or "fooo"
- "*" is equivalent to {0,}, "+" is equivalent to {1,}, "?" is equivalent to {0,1}

# Alternation

- |
  - You can use alternation to match a single regular expression out of several possible regular expressions
  - The alternation operator has the lowest precedence of all regex operators
  - That is, it tells the regex engine to match either everything to the left of the vertical bar, or everything to the right of the vertical bar
  - If you want to limit the reach of the alternation, you will need to use round brackets for grouping.
  - Ex. (cat|dog)

# Anchors

- Anchors do not match any character at all. Instead, they match a position before, after or between characters
  - ^: matches the position before the first character in the string
  - $: matches right after the last character in the string

# Single Quoting Regex

- Since many of the special characters used in regexs also have special meaning to the shell, it's a good idea to get in the habit of single quoting your regexs
- This will protect any special characters from being operated on by the shell

- Even though we are single quoting our regexs so the shell won't interpret the special characters, sometimes we still want to use an operator as itself
- To do this, we escape the character with a ¥ (backslash)

# Regex Metacharacters

- ¥b matches a word boundary, that is, the position between a word and a space
- ¥B matches a nonword boundary
- ¥d matches a digit character. Equivalent to [0–9].
- ¥D matches a nondigit character. Equivalent to [^0–9].
- ¥f matches a form-feed character
- ¥n matches a newline character
- ¥r matches a carriage return character

# Regex Metacharacters (cont'd)

- ¥s matches any white space including space, tab, form-feed
- ¥S matches any nonwhite space character
- ¥t matches a tab character
- ¥v matches a vertical tab character
- ¥w matches any word character including underscore. Equivalent to [A–Za–z0–9_].
- ¥W matches any nonword character

# Regular Expression Examples

# Regular Expression Example (1)

AP.[RL][ET]

match or unmatch ?

- APPLE ○
- APPLET ○
- APEAK ×
- APART ○

# Regular Expression Example (2)

APP[A–Z][A–Z][LR]

match or unmatch ?

- APPEAR ○
- APPLET ×
- APPEAL ○
- APPEND ×

# Regular Expression Example (3)

AP.+E

match or unmatch ?

- APPLE ○
- APPOINT ×
- APPAREL ○
- APEX ×

# Regular Expression Example (4)

AP{1,2}.*E

match or unmatch ?

- APPLE ○
- APPOINTMENT ○
- APPAREL ○
- APEX ○

# Regular Expression Example (5)

APP?A?R*[EL]

match or unmatch ?

- APPLE ○
- APART ×
- APPAREL ○
- APEX ○

# Regular Expression Example (6)

ˆAP.*EX?$

match or unmatch ?

- APPLE ○
- APPOINT ×
- APPAREL ×
- APEX ○

# Regular Expression Example (7)

¥w+¥d*¥w

match or unmatch ?

- LEVIN ○
- AE86 ○
- FD3S ○
- 320i ○

# You can try with …

- RegEx Pal
  https://www.regexpal.com/
- 正規表現チェッカー
  https://www-creators.com/tool/regex-checker
- Regular Expression Test Drive
  https://regex-testdrive.com/ja/dotest
- REGEXPER (not a checker)
  https://regexper.com/

# That's it today

## Assignment #2 in next week

今日はここまで

次週，課題2を出す