### Searching in a Library Catalogue

- To find articles, books, and other library holdings, a library generally provides a **database** containing information on its holdings.
- **OSCAR is the database frontend** providing access to the library database at OSU.
- **OSCAR makes it possible to search for the occurrence of literal strings** occurring in the author, title, keywords, call number, etc. associated with an item held by the library.

### Basic searching in OSCAR

- **Literal strings are composed of characters which naturally must be in the same character encoding system** (e.g. ASCII, ISO8859-1, UTF-8) as the strings encoded in the database.
- For literal strings, **OSCAR does not distinguish between upper and lower-case letters** (i.e. they aren’t so literal after all ;)
- **Adjacent words are searched as a phrase.**
  - art therapy
  - vitamin c
- In addition to querying literal strings, the **query language** of OSCAR also supports the use of:
  - special characters to abbreviate multiple options
  - special operators for combining two query strings (boolean operators) or modifying the meaning of a single string (unary operators)

### OSCAR: Special characters

- Use * for 1–5 characters at end or within a word.
  - art* finds arts, artists, artistic
  - gentle*n
- Use ** for any number of characters at end of word.
  - art** finds artificial, artillery
- Use ? for a single character at end or within a word.
  - gentles?
- The special * and ? characters must have at least 2 characters to their left. (∴ for efficiency reasons)

### OSCAR: Literal Strings and Operators (I)

- Use and or or to specify multiple words in any field, any order.
  - art and therapy
  - art or therapy
  - c+ or c++
- Use and not to exclude words.
  - art and not therapy

### OSCAR: Operators (II)

- Use parentheses to group words together when using more than one operator.
  - art therapy and not ((music or dance) therapy)
- Use near to specify words within 10 words of each other, in any order.
  - art near therapy
- Use within n to specify words within n words of each other. The value of n has no limit.
  - art within 12 therapy
Search engines

- Search engines (e.g., Google)
  - store a copy of all web pages
  - create an index to provide efficient access to this large number of pages (e.g., Google currently searches over 4 billion pages)
  - compute a rank for each web page to be able to rank the query results
- Search engines differ in various ways:
  - stemming: treat bird and birds as the same or not
capitalization: treat trip and Trip the same or not
- use of operators
- special interface for advanced searching
- how search results are ranked
- clustering: group similar results or not

Google: Operators (I)

- : Require a word to occur in the result
e.g., To find a restaurant that serves both tofu and BBQ one could try to+a+tofu+baq.
- : Disallow a word from occurring in the result
e.g., As a pastas purist, I search for pastas -pastatoes.
- : Include synonyms of the word
Quotation Marks (phrases)
e.g., “What Cheer” when looking for sites on What Cheer, Iowa

Google: Operators (II)

- intitle: Find words used in a title
e.g., intitle:Buckeye finds only web pages which has this word in the title
- inurl: Find words used in the url
e.g., inurl:lingo returns more linguistics webpages than doing
- link: Find pages that link to a certain page
e.g., link:www.osu.edu to show pages linking to the main osu web page
- site: Find pages that are part of a single domain
  e.g., I want to find strange attractions involving fish. Knowing one site which has such stuff, one can try fish site:www.roadsideamerica.com.

Google: Advanced searching

More elaborate web forms are provided as alternative to using operators:

- match all: matches all terms in your query
- match any: matches as many terms in your query as it can find
e.g., I'm looking for a restaurant that has bbq or bb-que or barbeque in the title

⇒ most search engines return “match all” followed by “match any” results

- exclude: eliminate documents which contain certain words
Information used to rank results

- Counting the number of links to and from a page, to determine how popular a page is. (As a result, unpopular or new pages require a more specific query to be found.)
- Keeping track of the nature of links to a page; linked pages might be thematically related. E.g., even if I never mention Sinclair Lewis on a page describing his book Babbit, it can be identified if many Sinclair Lewis sites link to my page.
- Bonuses/penalties for sites known to be of high/low quality
- Looking for keywords in metadata
- Counting how often a web result was clicked on by a user (click-through measurement)
- Various secret ingredients

Evaluating search results

What measures can one use to evaluate how successful a query is?

- Precision: How many of the pages returned are the ones we want?
  - E.g., Google gives me 400 hits for a query, 200 of which are related to the topic I want; precision = 50%.
- Recall: How many pages on the topic were actually given? (Hard to calculate for web searchers. E.g., Google gave me 200 pages I wanted, but there were actually 1000 pages on that topic out there somewhere on the internet; recall = 20%.

We saw earlier how to use our initial results to refine our query and improve precision.

Regular expressions: What they are

- A regular expression is a compact description of a set of strings, i.e., a language (in formal language theory).
- They can be used to search for occurrences of these strings.
- Regular expressions can only describe so-called regular languages.
- This means that some patterns cannot be specified using regular expressions, e.g., finding a string containing any number of as followed by exactly the same number of bs.
- Note that just like any other formalism, regular expressions as such have no linguistic contents, but they can be used to refer to strings encoding a natural language text.

Regular expressions: Tools that use them

- A variety of Unix tools (grep, sed, . . .), editors (emacs, . . .), and programming languages (perl, python, . . .) incorporate regular expressions.
- Implementations are very efficient so that large text files can be searched quickly, but not efficient enough for web searching → no web search engine offers them (yet).
- The various tools and languages differ w.r.t. the exact syntax of the regular expressions they allow.

The syntax of regular expressions (I)

Regular expressions consist of:

- Strings of literal characters: c, A100, natural language, 30 years!
- Disjunction: ordinary disjunction: devoured, ate, family\{ies\}
- Character classes: [tT]he, be\{co\}me
- Ranges: [A-Z] (any capital letter)
- Negation: [^a] (any symbol but a)\[^A-Z\-9\] (not an uppercase letter or number)

The syntax of regular expressions (II)

- Counters:
  - Optional?: ?
  - Colour: \^\?
  - Any number of occurrences: * (Kleene star)\[^0-9\]*\n  - At least one occurrence: +\[^0-9\]+\n  - Wildcard for any character: .
  - beg, n for any character in between beg and n

The syntax of regular expressions (III)

- Escaped characters: to specify a character with a special meaning (*, +, ?, (, ), |, [ ] ) it is preceded by a backslash (\)
  - E.g., a period is expressed as .
- Operator precedence, from highest to lowest:
  - Parentheses ()
  - Counters * + ?
  - Character sequences \^\?

Grep

- Grep is a powerful and efficient program for searching in text files using regular expressions.
- It is standard on Unix, Linux, and Mac OSX, and there also are various ports to Windows (e.g., http://gnuwin32.sourceforge.net/packages/grep.html, http://www.interlog.com/~tcharron/grep.html or http://www.wingrep.com/).
- The version of grep that supports the full set of operators mentioned above is generally called egrep (for extended grep).
In the following, we assume a text file `f.txt` containing, among others, the strings that we mention as matching.

- **Strings of literal characters:**
  - `grep 'and' f.txt` matches *and*
  - `grep 'Ayn Rand, Candy' f.txt` and so on

- **Character classes:**
  - `grep '[0-9]' f.txt` matches a year
  - `grep '[0-9][0-9][0-9][0-9]' f.txt` matches the year 1776, the year 1812, the year 2001, and so on

- **Escaped characters:**
  - `grep 'why?' f.txt` matches `why?` whereas `grep 'why' f.txt` matches `why` and `wh`

**Corpora**

- *A corpus* is a collection of text.
- Corpora with the works of various writers, newspaper texts, etc. have been collected and electronically encoded.
- Corpora can be quite large
- The British National Corpus is a 100 million word collection representing a wide cross-section of current written and spoken British English.

**How corpora can be searched**

- Both the BNC and the European Parliament corpus can be searched using on-line web-forms.
- Both of the web forms allow **regular expressions** for advanced searching.
- To provide efficient searching in large corpora, in these search engines regular expressions over characters are limited to single tokens (i.e. generally words).
- **BNC:**
  - web form: [http://sara.natcorp.ox.ac.uk/lookup.html](http://sara.natcorp.ox.ac.uk/lookup.html)
  - regular expressions are enclosed in `{` `}`

- **European Parliament Corpus:**
  - web form: [http://logos.uio.no/cgi-bin/opus/opuscp.pl?corpus=EUROPARL.lang=en](http://logos.uio.no/cgi-bin/opus/opuscp.pl?corpus=EUROPARL.lang=en)
  - in the simplest case, regular expressions are enclosed in " "

**Grep: Examples for using regular expressions**

- **disjunction (`|`):**
  - `egrep 'G|g' f.txt` matches `G` or `g`, so `egrep 'G|gouda' f.txt` matches `gouda` or `Gouda`. Note that `(G|g)ouda` has the same effect.

- **grouping with parentheses:**
  - `egrep 'un(terest|e|xciting)' f.txt` matches uninteresting or unexciting.

- **Any character (`.`):**
  - `egrep 'o.e' f.txt` matches `ore`, `one`, `ole`