ΜΥΕ003: Ανάκτηση Πληροφορίας

Διδάσκουσα: Ευαγγελία Πιτουρά

Εισαγωγή στη Lucene. Περιγραφή Εργασίας.

Ακαδημαϊκό Έτος 2023-2024

Εργασία

Θέμα: Σχεδιασμός και υλοποίηση ενός συστήματος αναζήτησης πληροφορίας σχετικά με επιστημονικά άρθρα.

Βήμα 1: Δημιουργία συλλογής (corpus) από σχετικά άρθρα.

Βήμα 2: Υλοποίηση μιας μηχανή αναζήτησης αυτών των άρθρων.
Συγκεκριμένα:

- Ο χρήστης θα θέτει ερωτήματα.
- Το σύστημα θα επιστρέφει τα συναφή με το ερώτημα άρθρα της συλλογής σας σε διάταξη με βάση τη συνάφεια τους με το ερώτημα.

Για την υλοποίηση, θα χρησιμοποιήστε τη βιβλιοθήκη **Lucene**

Προαιρετικό ερώτημα: Επέκταση της αναζήτησης με σημασιολογική ανάκτηση με χρήση LLM (Θα ανακοινωθεί σύντομα)

Διαδικαστικά

Καταληκτικές Ημερομηνίες

- Τετάρτη 17 Απριλίου 2024: Σύντομη περιγραφή τους σχεδιασμού και της συλλογής των δεδομένων
- Τετάρτη 22 Μαΐου 2024: Παράδοση εργασίας
- Εβδομάδα 27 Μάϊου (Τρίτη, Τετάρτη): Προφορική εξέταση εργασίας

Οι καταληκτικές ημερομηνίες είναι αυστηρές, **δεν** γίνονται δεκτές αργοπορημένες παραδόσεις ασκήσεων

- Παράδοση μέσω ecourse
- Τελική εργασία στο github

- Η εργασία μπορεί να γίνει σε ομάδες έως 2 ατόμων.
- Η εργασία μετράει σε ποσοστό 50% στο βαθμό σας στο μάθημα.

Περιεχόμενα Παρουσίασης

Σύντομη παρουσίαση

- Ευρετήρια ζώνης
- Lucene
- Εργασία
- Περίληψη αποτελεσμάτων

ΜΥΕ003: Ανάκτηση Πληροφορίας
Ευρετήρια ζώνης

Διδάσκουσα: Ευαγγελία Πιτουρά

Ακαδημαϊκό Έτος 2023-2024

Παραμετρικά ευρετήρια και ευρετήρια ζώνης

- *Για αρκετά έγγραφα εκτός του κειμένου, επιπρόσθετη πληροφορία είναι χωρισμένα σε τμήματα* με διαφορετική σημασία:
  - Συγγραφέας
  - Τίτλος
  - Ημερομηνία δημοσίευσης
  - Γλώσσα
  - κλπ

- Καλούνται και μεταδεδομένα (metadata) του εγγράφου

Παραμετρικά ερωτήματα

- Συχνά αναζήτηση με βάση τα μεταδεδομένα
  - Π.χ., βρες όλα τα έγγραφα που έγγραψε ο William Shakespeare το 1601, που περιέχουν τις λέξεις *alas poor Yorick*
  - Year = 1601 είναι παράδειγμα ενός πεδίου (field)
  - Επίσης, author last name = shakespeare, κλπ
- Ερωτήματα με πεδία (παραμετρικά ερωτήματα) συνήθως ερμηνεύονται ως *συζευκτικά* (conjunction, σύνδεση με AND) πρέπει να ισχύουν όλα

Παραμετρική αναζήτηση

Ζώνη

Επίσης, συχνά τα *έγγραφα είναι χωρισμένα σε τμήματα (ζώνες)* με διαφορετική σημασία:

- Η ζώνη (zone) είναι μια περιοχή ενός εγγράφου που περιέχει κείμενο, π.χ.,
  - Title (τίτλος)
  - Abstract (περίληψη)
  - References (αναφορές) …
- Πρέπει να τροποποιήσουμε τα ευρετήρια ώστε να επιτρέψουμε σχετικά ερωτήματα όπως
  - πχ, βρες έγγραφα με τον όρο «merchant» στον τίτλο τους

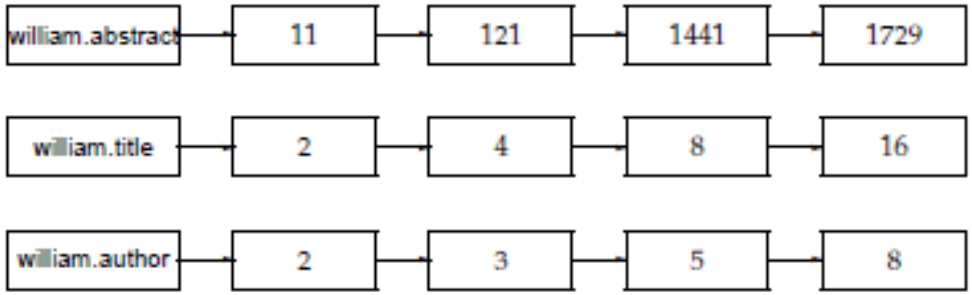
- Επίσης χρήσιμα αν θέλουμε να δώσουμε *μεγαλύτερο βάρος* σε εμφανίσεις όρων στον τίτλο ή στην περίληψη

Ευρετήριο πεδίου

- Ευρετήριο πεδίου (Field index) ή παραμετρικό ευρετήριο (parametric index): καταχωρήσεις (postings) για κάθε πεδίο
  - Συχνά ειδικού τύπου (πχ δέντρα διαστήματος για ημερομηνίες)

Βασικό ευρετήριο ζώνης encoded στο λεξικό (διαφορετικές λίστες καταχωρήσεων)

Ένα ευρετήριο για κάθε ζώνη/πεδίο:



Επέκταση καταχωρήσεων

Ένα ενιαίο ευρετήριο για κάθε έγγραφο

Η πληροφορία ζώνης στις λίστες καταχώρησης:

- Ευρετήριο πεδίου: καλύτερο για παραμετρικά ερωτήματα
- Επέκταση καταχωρήσεων: ποιο αποδοτικό για υπολογισμό «ενιαίας» ζυγισμένης συνάφειας

Lucene

Εισαγωγή

- Open source search software
- Lucene Core provides Java-based indexing and search as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities
- Let you add search to your application, not a complete search system by itself -- software library not an application
- Written by Doug Cutting

Εισαγωγή

- An “engine” used by LinkedIn, Twitter, Netflix, Oracle, …
  - and many more (see http://wiki.apache.org/lucene-java/PoweredBy)

- Ports/integrations to other languages
  - C/C++, C#, Ruby, Perl, PHP

  - PyLucene: a Python port of the Core project
    - Allows use of Lucene's text indexing and searching capabilities from Python.

    https://lucene.apache.org/pylucene/

Μπορείτε να την κατεβάσετε από

http://lucene.apache.org/core/

Some features (indexing)

Scalable, high-performance indexing

- over 800GB/hour on modern hardware
- small RAM requirements -- only 1MB heap
- incremental indexing as fast as batch indexing
- index size roughly 20-30% the size of text indexed

Some features (search)

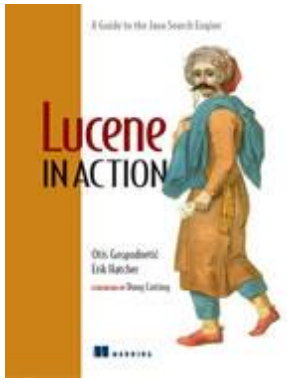
Powerful, accurate and efficient search algorithms

- *ranked* searching -- best results returned first
- many powerful *query types*: phrase queries, wildcard queries, proximity queries, range queries and more
- *fielded searching* (e.g. title, author, contents)
- *nearest-neighbor search* for high-dimensionality vectors
- sorting by any field
- multiple-index searching with merged results
- allows simultaneous update and searching
- flexible faceting, highlighting, joins and result grouping
- fast, memory-efficient and typo-tolerant suggesters
- pluggable ranking models, including the Vector Space Model and Okapi BM25
- configurable storage engine (codecs)

Στόχος της παρουσίασης: σύντομη εισαγωγή

Περισσότερες πληροφορίες

https://lucene.apache.org/core/9_10_0/index.html



https://www.manning.com/books/lucene-in-action-second-edition

- Lucene tutorials

http://www.lucenetutorial.com/

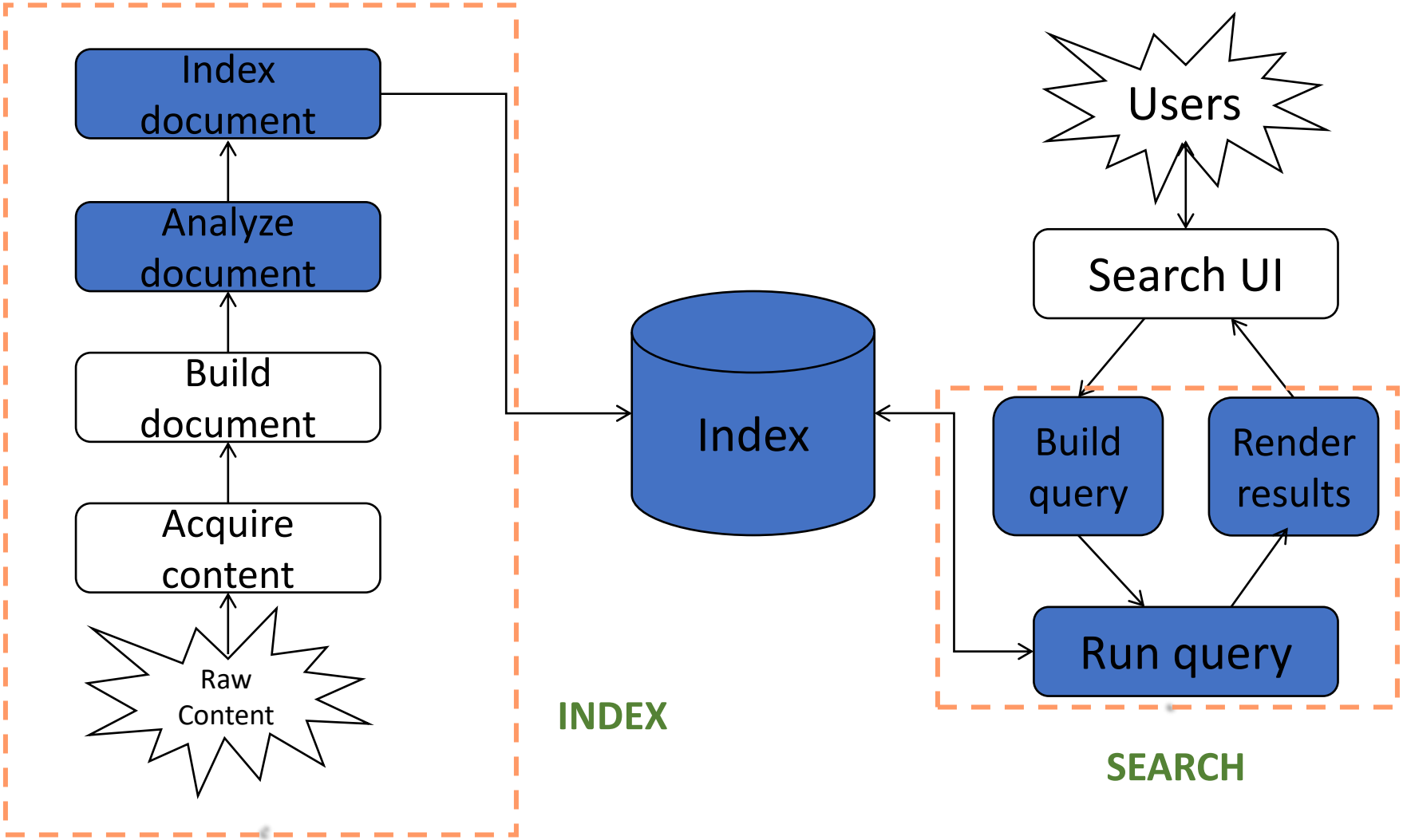
- Exampled updated to 9.x

https://www.lucenetutorial.com/lucene-in-5-minutes.html

- Lucene demo

https://lucene.apache.org/core/9_10_0/demo/index.html

Βασικές έννοιες



Βασικές έννοιες: document

- The unit of search and index.
- Indexing involves adding Documents to an IndexWriter.
- Searching involves retrieving Documents from an index via an IndexSearcher.

- A document consists of one or more Fields
  - A Field is a name-value pair.
    example: title, body, or metadata (creation time, etc)

Βασικές έννοιες: Fields

- You have to translate raw content into Fields
- Search a field using <field-name:term>,
  - e.g., title:lucene

Βασικές έννοιες: index

- Indexing in Lucene
  1. Create documents comprising of one or more Fields
  2. Add these Documents to an IndexWriter.

Βασικές έννοιες: search

Searching requires an index to have already been built.

- It involves
  1. Create a Query (usually via a QueryParser) and
  2. Handle this Query to an IndexSearcher, which returns a list of Hits.

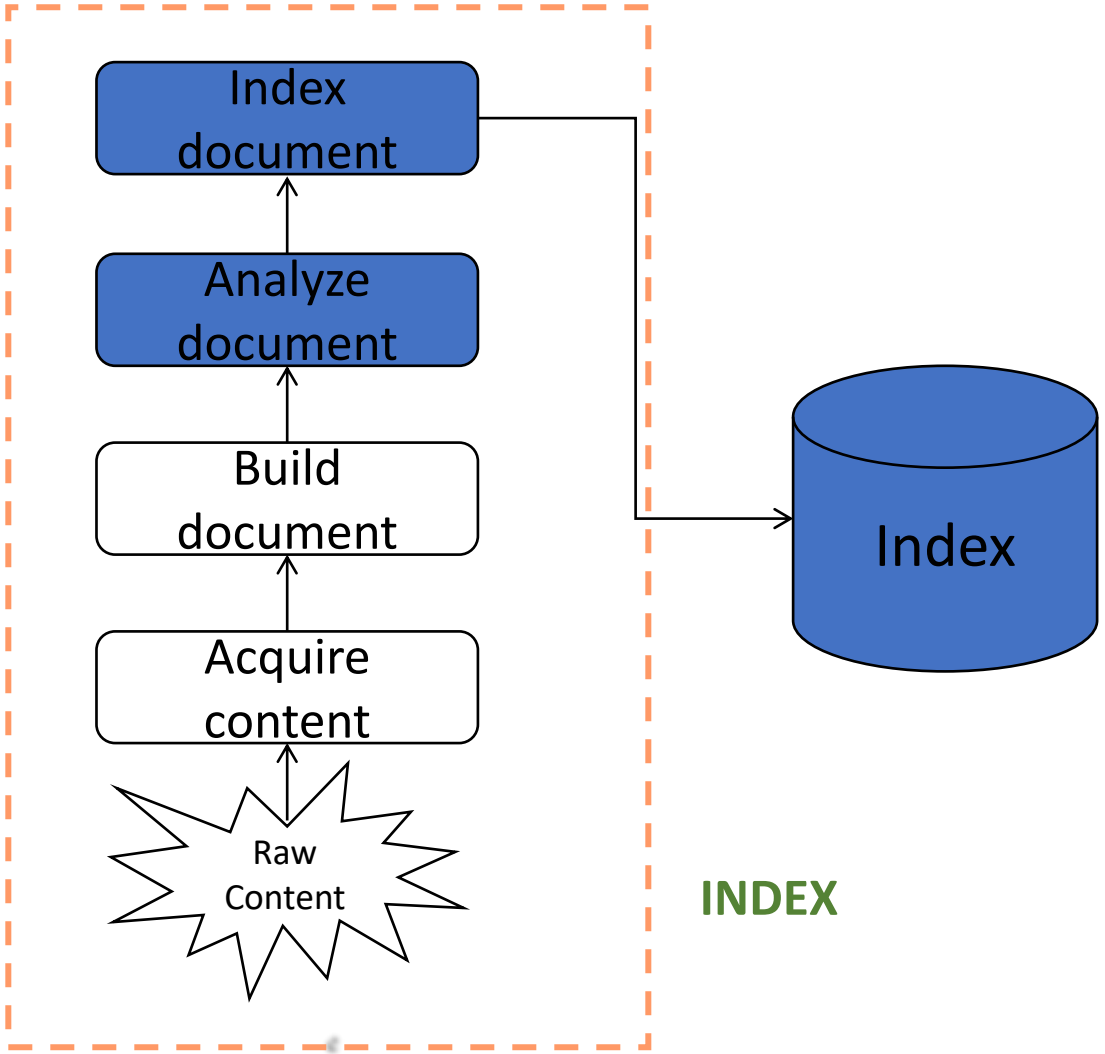
- The Lucene query language allows the user to specify
  - which field(s) to search on,
  - which fields to give more weight to (boosting),
  - the ability to perform boolean queries (AND, OR, NOT) and
  - other functionality.

Lucene in a search system:
index

Lucene in a search system: index

Steps

1. Acquire content
2. Build document
3. Analyze document
4. Index documents



Step 1: Acquire and build content

Not supported by core Lucid

Collection depending on type may require:

- Crawler or spiders (web)
- Specific APIs provided by the application (e.g., Twitter, FourSquare, imdb)
- Scrapping https://pypi.org/project/beautifulsoup4/
- Complex software if scattered at various location, etc

Complex documents (e.g., XML, JSON, relational databases, pptx etc)

Tika the Apache Tika™ toolkit detects and extracts metadata and text from over a thousand different file types (such as PPT, XLS, and PDF)
http://tika.apache.org/

Step 1: Acquire and build content

OpenNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, language detection, chunking (extracting sentences from unstructured text), parsing, and coreference resolution (find all expressions that refer to the same entity in the text)

https://opennlp.apache.org/

Step 2:Build Documents

Create documents by adding fields

Fields may be

- indexed or not
  - Indexed fields may or may not be analyzed (i.e., tokenized with an `Analyzer`)
    - *Non-analyzed fields view the entire value as a single token* (useful for URLs, paths, dates, social security numbers, ...)
- stored or not
  - Useful for fields that you'd like to display to users
- Optionally store term vectors and other options such as positional indexes

Step 2:Build Documents

Create documents by adding fields

Step 1 – Create a method to get a Lucene document from a text file.

Step 2 – Create various fields which are key value pairs containing keys as names and values as contents to be indexed.

Step 3 – Set field to be analyzed or not, stored or not

Step 4 – Add the newly-created fields to the document object and return it to the caller method.

Step 2:Build Documents (παλιότερη version)

```
private Document getDocument(File file) throws IOException {
Document document = new Document();

 //index file contents
Field contentField = new Field(LuceneConstants.CONTENTS,
new FileReader(file))
//index file name
Field fileNameField = new Field(LuceneConstants.FILE_NAME, file.getName(), Field.Store.YES,Field.Index.NOT_ANALYZED);

//index file path
 Field filePathField = new Field(LuceneConstants.FILE_PATH, file.getCanonicalPath(),  Field.Store.YES,Field.Index.NOT_ANALYZED);

document.add(contentField);
document.add(fileNameField);
 document.add(filePathField);

return document;
}
```

Step 3:analyze and index

Create an IndexWriter and add documents to it with addDocument();

Core indexing classes

- Analyzer
  - Extracts tokens from a text stream
- IndexWriter
  - create a new index, open an existing index, and
  - add, remove, or update documents in an index
- Directory
  - Abstract class that represents the location of an index

παλιότερη version

```
Analyzer analyzer = new StandardAnalyzer();

// INDEX: Store the index in memory: (για την εργασία θα το αποθηκεύστε στο δίσκο – θα δημιουργηθεί μια φορά στην αρχή)
Directory directory = new RAMDirectory();
// To store an index on disk, use this instead:
//  Directory directory = FSDirectory.open("/tmp/testindex");
IndexWriterConfig config = new IndexWriterConfig(analyzer);
IndexWriter iwriter = new IndexWriter(directory, config);
Document doc = new Document();
String text = "This is the text to be indexed.";
doc.add(new Field("fieldname", text, TextField.TYPE_STORED));
iwriter.addDocument(doc);
iwriter.close();

// SEARCH: Now search the index:
  DirectoryReader ireader = DirectoryReader.open(directory);
  IndexSearcher isearcher = new IndexSearcher(ireader);
   // Parse a simple query that searches for "text":
   QueryParser parser = new QueryParser("fieldname", analyzer);
   Query query = parser.parse("text");
   ScoreDoc[] hits = isearcher.search(query, null, 1000).scoreDocs;
// Iterate through the results:
   for (int i = 0; i < hits.length; i++) {
    Document hitDoc = isearcher.doc(hits[i].doc);
}
   ireader.close();
   directory.close();
```

Using `Field` options

Index	Store	TermVector	Example usage
NOT_ANALYZED	YES	NO	Identifiers, telephone/SSNs, URLs, dates, ...
ANALYZED	YES	WITH_POSITIONS_OFFSETS	Title, abstract
ANALYZED	NO	WITH_POSITIONS_OFFSETS	Body
NO	YES	NO	Document type, DB keys (if not used for searching)
NOT_ANALYZED	NO	NO	Hidden keywords

Analyzers

Tokenizes the input text

- Common Analyzers
  - WhitespaceAnalyzer
    Splits tokens on whitespace
  - SimpleAnalyzer
    Splits tokens on non-letters, and then lowercases
  - StopAnalyzer
    Same as SimpleAnalyzer, but also removes stop words
  - StandardAnalyzer
    Most sophisticated analyzer that knows about certain token types, lowercases, removes stop words, ...

Analysis examples

"The quick brown fox jumped over the lazy dog"

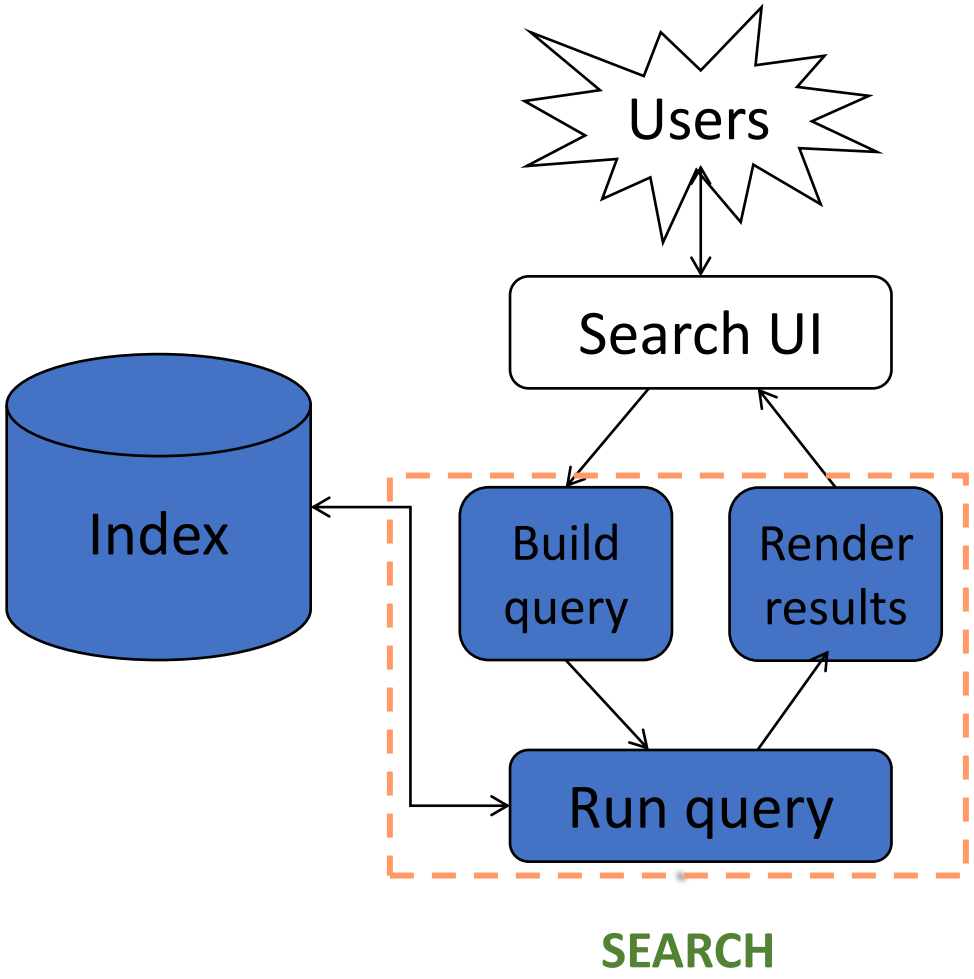
- `WhitespaceAnalyzer`
  - [The] [quick] [brown] [fox] [jumped] [over] [the] [lazy] [dog]
- `SimpleAnalyzer`
  - [the] [quick] [brown] [fox] [jumped] [over] [the] [lazy] [dog]
- `StopAnalyzer`
  - [quick] [brown] [fox] [jumped] [over] [lazy] [dog]
- `StandardAnalyzer`
  - [quick] [brown] [fox] [jumped] [over] [lazy] [dog]

More analysis examples

- “XY&Z Corporation – xyz@example.com”
- `WhitespaceAnalyzer`
  - [XY&Z] [Corporation] [-] [xyz@example.com]
- `SimpleAnalyzer`
  - [xy] [z] [corporation] [xyz] [example] [com]
- `StopAnalyzer`
  - [xy] [z] [corporation] [xyz] [example] [com]
- `StandardAnalyzer`
  - [xy&z] [corporation] [xyz@example.com]

Lucene in a search system: search

Lucene in a search system: search



Search User Interface (UI)

No default search UI, but many useful modules

General instructions

- Simple (do not present a lot of options in the first page)
  search box better than 2-step process
- Result presentation is very important
  - highlight matches
  - make sort order clear, etc

Core searching classes

- QueryParser
  - Parses a textual representation of a query into a Query instance
  - Constructed with an analyzer used to interpret query text in the same way as the documents are interpreted
- Query
  - Contains the results from the QueryParser which is passed to the searcher
  - Abstract query class
  - Concrete subclasses represent specific types of queries, e.g., matching terms in fields, boolean queries, phrase queries, …
- IndexSearcher
  - Central class that exposes several search methods on an index
  - Returns TopDocs with max n hits

παλιότερη version

```
Analyzer analyzer = new StandardAnalyzer();

//INDEX:  Store the index in memory: (για την εργασία θα το αποθηκεύστε στο δίσκο – θα δημιουργηθεί μια φορά στην αρχή))
Directory directory = new RAMDirectory();
// To store an index on disk, use this instead:
//  Directory directory = FSDirectory.open("/tmp/testindex");
IndexWriterConfig config = new IndexWriterConfig(analyzer);
IndexWriter iwriter = new IndexWriter(directory, config);
Document doc = new Document();
String text = "This is the text to be indexed.";
doc.add(new Field("fieldname", text, TextField.TYPE_STORED));
iwriter.addDocument(doc);
iwriter.close();

// QUERY: Now search the index:
  DirectoryReader ireader = DirectoryReader.open(directory);
  IndexSearcher isearcher = new IndexSearcher(ireader);
   // Parse a simple query that searches for "text":
   QueryParser parser = new QueryParser("fieldname", analyzer);
   Query query = parser.parse("text");
   ScoreDoc[] hits = isearcher.search(query, null, 1000).scoreDocs;
// Iterate through the results:
   for (int i = 0; i < hits.length; i++) {
    Document hitDoc = isearcher.doc(hits[i].doc);
}
   ireader.close();
   directory.close();
```

QueryParser syntax examples

Query expression	Document matches if…
java	Contains the term *java* in the default field
java junit java OR junit	Contains the term *java* or *junit* or both in the default field (*the default operator can be changed to* AND)
+java +junit java AND junit	Contains both *java* and *junit* in the default field
title:ant	Contains the term *ant* in the title field
title:extreme - subject:sports	Contains *extreme* in the title and not *sports* in subject
(agile OR extreme) AND java	Boolean expression matches
title:”junit in action”	Phrase matches in title
title:”junit action”~5	Proximity matches (within 5) in title
java*	Wildcard matches
java~	Fuzzy matches
lastmodified:[1/1/09 TO 12/31/09]	Range matches

Scoring

- Scoring function uses basic *tf-idf* scoring with
  - Programmable boost values for certain fields in documents
  - Length normalization
  - Boosts for documents containing more of the query terms
- IndexSearcher provides a method that explains the scoring of a document

Summary

To use Lucene

1. Create Documents by adding Fields;
2. Create an IndexWriter and add documents to it with addDocument();

3. Call QueryParser.parse() to build a query from a string; and
4. Create an IndexSearcher and pass the query to its search() method.

Summary: Lucene API packages

https://lucene.apache.org/core/9_10_0/core/index.html

The Lucene API is divided into several packages:

- org.apache.lucene.analysis defines an abstract Analyzer API for converting text from a **Reader** into a **TokenStream**, an enumeration of token Attributes. A TokenStream can be composed by applying TokenFilters to the output of a Tokenizer. Tokenizers and TokenFilters are strung together and applied with an Analyzer. analysis-common provides a **number of Analyzer** implementations, including StopAnalyzer and the grammar-based StandardAnalyzer.
- org.apache.lucene.codecs provides an abstraction over the encoding and decoding of the inverted index structure, as well as different implementations that can be chosen depending upon application needs.
- org.apache.lucene.document provides a simple Document class. A Document is simply **a set of named Fields**, whose values may be strings or instances of Reader.
- org.apache.lucene.index provides two primary classes: IndexWriter, which creates and adds documents to indices; and IndexReader, which accesses the data in the index.

Summary: Lucene API packages

https://lucene.apache.org/core/9_10_0/core/index.html

- org.apache.lucene.search provides data structures to represent queries (ie **TermQuery** for individual words, **PhraseQuery** for phrases, and **BooleanQuery** for boolean combinations of queries) and the **IndexSearcher** which turns queries into TopDocs. A number of QueryParsers are provided for producing query structures from strings or xml.
- org.apache.lucene.store defines an abstract class for storing persistent data, the **Directory**, which is a collection of named files written by an IndexOutput and read by an IndexInput. Multiple implementations are provided, but FSDirectory is generally recommended as it tries to use operating system disk buffer caches efficiently.
- org.apache.lucene.util contains a few handy data structures and util classes, ie FixedBitSet and PriorityQueue.lucene.analysis defines an abstract Analyzer API for converting text from a Reader into a TokenStream, an enumeration of token Attributes.

https://solr.apache.org/

Lucene is a full-text search engine library, whereas Solr is a full-text search engine web application built on Lucene

Elasticsearch

https://www.elastic.co/

- Built on top of Lucene.
- ML functionality
- A distributed system/search engine for scaling horizontally
- Provides other features like thread-pool, queues, node/cluster monitoring API, data monitoring API, Cluster management, etc.
- Hosts data on data nodes. Each data node hosts one or more indices, and each index is divided into shards with each shard holding part of the index's data. Each shard created in Elasticsearch is a separate Lucene instance or process.

Λίγα περισσότερα για την εργασία

Εργασία

Θέμα: Σχεδιασμός και υλοποίηση ενός συστήματος αναζήτησης πληροφορίας σε επιστημονικά άρθρα

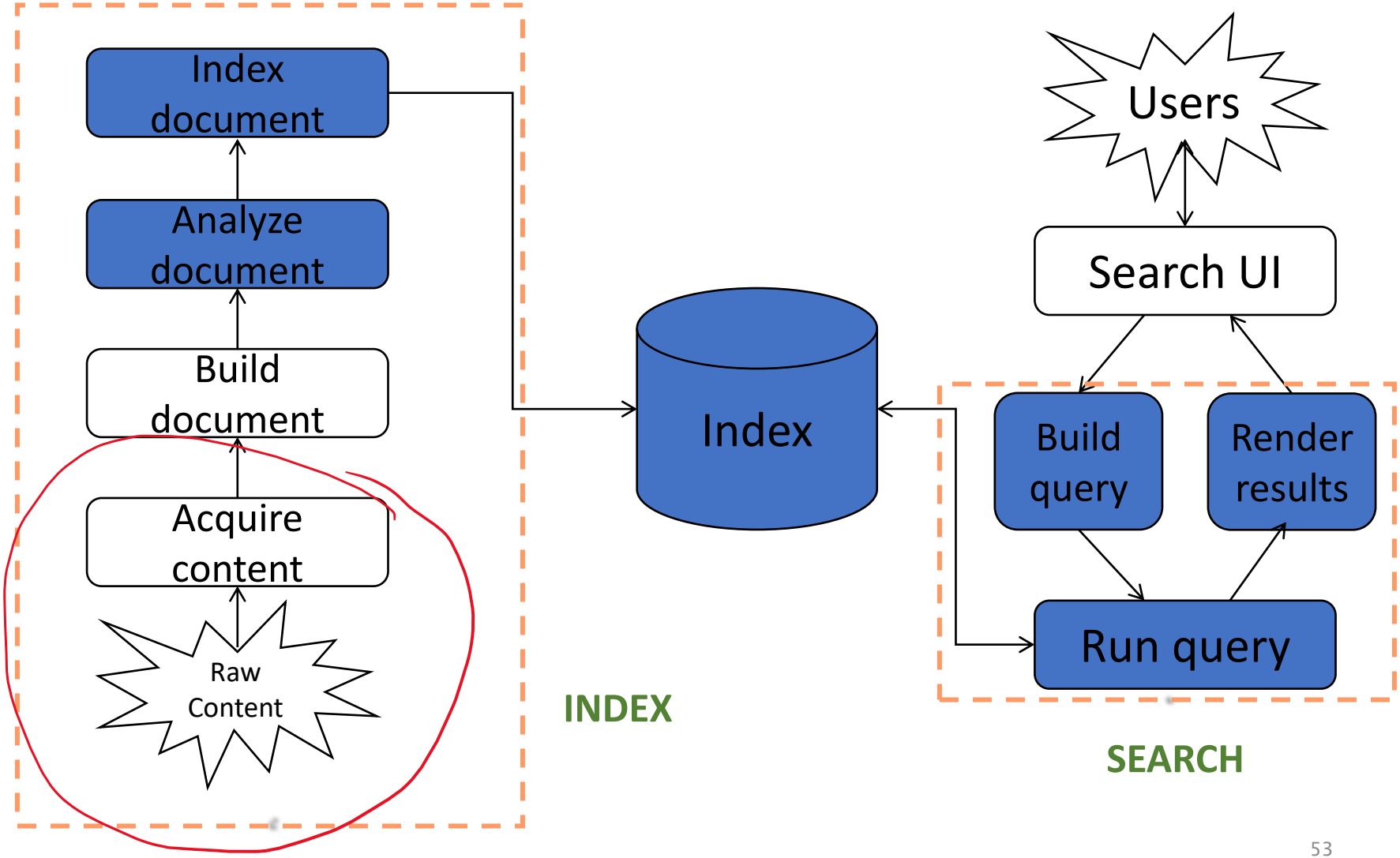
Βήμα 1: Δημιουργία συλλογής (corpus) από σχετικά άρθρα.

Βήμα 2: Υλοποίηση μιας μηχανή αναζήτησης αυτών των άρθρων.
Συγκεκριμένα:

- Ο χρήστης θα θέτει ερωτήματα.
- Το σύστημα θα επιστρέφει τα συναφή με το ερώτημα άρθρα της συλλογής σας σε διάταξη με βάση τη συνάφεια τους με το ερώτημα.

Για την υλοποίηση, θα χρησιμοποιήστε τη βιβλιοθήκη **Lucene**

Βασικές έννοιες



Συλλογή εγγράφων

Συλλογή εγγράφων (corpus). Αρχικά, πρέπει να κατεβάσετε τη συλλογή από το Kaggle ~10,000 άρθρα

Η συλλογή πρέπει να περιλαμβάνει *τουλάχιστον* 200 άρθρα

Kaggle

https://www.kaggle.com/datasets/rowhitswami/nips-papers-1987-2019-updated/data?select=papers.csv

αυτό είναι το βασικό

papers.csv

source_id

Unique ID of the published paper

year

Year in which it get published

title

Title of the published paper

abstract

Abstract of the published paper

full_text

Full text of the published paper

προαιρετικά

authors.csv

source_id

Unique ID of the published paper

first_name

First name of the author

last_name

Last name of the author

institution

Institution/Organization of the author

Εργασία

Ανάλυση κειμένου και κατασκευή ευρετηρίου. Η Lucene παρέχει τη δυνατότητα για stemming, απαλοιφή stop words, επέκταση συνωνύμων, κλπ.

Επίσης, κάποιες λειτουργίες, όπως η διόρθωση τυπογραφικών λαθών, ή η επέκταση ακρωνύμων, μπορούν να γίνουν εναλλακτικά κατά τη διάρκεια της αναζήτησης (τροποποιώντας το ερώτημα).

Επιλέξτε το είδος της ανάλυσης που θεωρείτε κατάλληλο και εξηγείστε την επιλογή σας.

Εργασία

Αναζήτηση

Το σύστημά σας θα πρέπει να υποστηρίζει
(α) αναζήτηση με λέξεις κλειδιά και
(β) αναζήτηση πεδίου, δηλαδή, την εμφάνιση όρων σε συγκεκριμένα πεδία (στον τίτλο, abstract, full text
(γ) έναν ακόμα τρόπο αναζήτησης της επιλογής σας

Εργασία

Αναζήτηση

Επίσης, να διατηρεί πληροφορία από την ιστορία των αναζητήσεων. Χρησιμοποιείστε αυτήν την πληροφορία για να προτείνετε εναλλακτικά ερωτήματα.

Προαιρετικό ερώτημα: συμπεριλάβατε στο έγγραφο και το όνομα του συγγραφέα και υποστηρίξτε αναζήτηση πεδίου και με το όνομα

Εργασία

Παρουσίαση Αποτελεσμάτων. Το σύστημα σας θα πρέπει να παρουσιάζει τα αποτελέσματα σε διάταξη με βάση τη συνάφεια τους με το ερώτημα.

Επιπρόσθετα, θα πρέπει
(1) Να παρουσιάζει τα αποτελέσματα ανά 10, με δυνατότητα στο χρήστη να προχωρήσει στα επόμενα.
(2) Οι λέξεις κλειδιά να παρουσιάζονται τονισμένες στο αποτέλεσμα.
(3) Να παρέχει δυνατότητα αναδιάταξης των αποτελεσμάτων με βάση τη χρονιά που εμφανίστηκε το άρθρο.

Εργασία: Προαιρετικό ερώτημα

Προαιρετικό Ερώτημα. Το σύστημα θα πρέπει να παρέχει τη δυνατότητα σημασιολογικής ανάκτησης (λεπτομερής εκφώνηση θα δοθεί τις επόμενες εβδομάδες).

Εργασία

Φάση 1:

Δύο στόχοι:

(1) Εξοικείωση με τη συλλογή

(2) Αρχικά βήματα υλοποίησης

(2α) Εγκατάσταση Lucene

(2b) Αρχικός σχεδιασμός

Τι θα παραδώσετε:

link στη github σελίδα που θα περιέχει

(1) Περιγραφή της συλλογής και του Document (ποια fields)

(2) Μια σύντομη (1-2 σελίδες) αρχική περιγραφή του συστήματος

Εργασία

Φάση 2:
Στόχος:
Ολοκλήρωση της εργασίας

Τι θα παραδώσετε
Στη github σελίδα:
Περιγραφή της εργασίας (κείμενο)
Πηγαίος κώδικας

ΜΥΕ003: Ανάκτηση Πληροφορίας
Παρουσίαση Αποτελεσμάτων

Διδάσκουσα: Ευαγγελία Πιτουρά

Ακαδημαϊκό Έτος 2023-2024

Περιλήψεις αποτελεσμάτων

- Αφού έχουμε διατάξει τα έγγραφα που ταιριάζουν με το ερώτημα, θέλουμε να τα παρουσιάσουμε στο χρήστη
- Πιο συχνά ως μια λίστα από τίτλους εγγράφων, URL, μαζί με μια μικρή περίληψη (**result snippet**), aka "10 blue links"

John McCain
John McCain 2008 - The Official Website of **John McCain**'s 2008 Campaign for President ... African American Coalition; Americans of Faith; American Indians for **McCain**; Americans with ...
www.**johnmccain**.com · Cached page

JohnMcCain.com - **McCain**-Palin 2008
John McCain 2008 - The Official Website of **John McCain**'s 2008 Campaign for President ... African American Coalition; Americans of Faith; American Indians for **McCain**; Americans with ...
www.**johnmccain**.com/Informing/Issues · Cached page

John McCain News- msnbc.com
Complete political coverage of **John McCain**. ... Republican leaders said Saturday that they were worried that Sen. **John McCain** was heading for defeat unless he brought stability to ...
www.msnbc.msn.com/id/16438320 · Cached page

John McCain | Facebook
Welcome to the official Facebook Page of **John McCain**. Get exclusive content and interact with **John McCain** right from Facebook. Join Facebook to create your own Page or to start ...
www.facebook.com/**johnmccain** · Cached page

Περιλήψεις αποτελεσμάτων

- Η περιγραφή του εγγράφου είναι *κρίσιμη* γιατί συχνά οι χρήστες βασίζονται σε αυτήν για να αποφασίσουν αν το έγγραφο είναι σχετικό
  - Δε χρειάζεται να διαλέξουν ένα-ένα τα έγγραφα με τη σειρά

Ο τίτλος αυτόματα από μεταδεδομένα, αλλά πώς να υπολογίσουμε τις περιλήψεις;

Περιλήψεις αποτελεσμάτων

Δύο βασικά είδη περιλήψεων

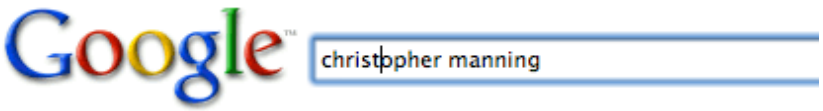
- Μια στατική περίληψη (static summary) ενός εγγράφου είναι πάντα η ίδια *ανεξάρτητα από το ερώτημα* που έθεσε ο χρήστης
- Μια δυναμική περίληψη (dynamic summary) εξαρτάται από το ερώτημα (query-dependent). Προσπαθεί να εξηγήσει γιατί το έγγραφο ανακτήθηκε για το *συγκεκριμένο* κάθε φορά ερώτημα

Στατικές Περιλήψεις

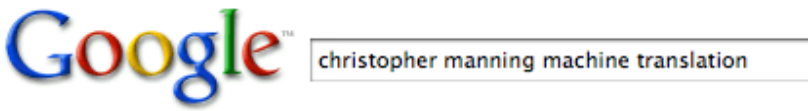
- Σε ένα τυπικό σύστημα η στατική περίληψη είναι ένα υποσύνολο του εγγράφου
- Απλός ευριστικός: οι *πρώτες περίπου 50 λέξεις* του εγγράφου
  - cached κατά τη δημιουργία του ευρετηρίου
- Πιο εξελιγμένες μέθοδοι (text summarization) – βρες από κάθε έγγραφο κάποιες σημαντικές προτάσεις
  - Απλή γλωσσολογική επεξεργασία (NLP) με ευριστικά για να βαθμολογηθεί κάθε πρόταση (πληροφορία θέσης: πρώτη και τελευταία παράγραφος, πρώτη και τελευταία πρόταση στην παράγραφο, και περιεχομένου: σημαντικές λέξεις)
  - Η περίληψη αποτελείται από τις προτάσεις με το μεγαλύτερο βαθμό
- Ή και πιο περίπλοκη γλωσσολογική επεξεργασία για τη σύνθεση/δημιουργία περίληψης

Δυναμικές Περιλήψεις

- Παρουσίασε ένα ή περισσότερα *«παράθυρα»* (windows, snippets) μέσα στο έγγραφο που να περιέχουν αρκετούς από τους όρους του ερωτήματος
  - *"KWIC" snippets*: αναπαράσταση Keyword-in-Context



Christopher Manning, Stanford NLP
Christopher Manning, Associate Professor of Computer Science and Linguistics, Stanford University.
nlp.stanford.edu/~**manning**/ - 12k - Cached - Similar pages



Christopher Manning, Stanford NLP
Christopher Manning, Associate Professor of Computer Science and Linguistics, ...
computational semantics, **machine translation**, grammar induction, ...
nlp.stanford.edu/~**manning**/ - 12k - Cached - Similar pages

Christopher Manning, Stanford NLP
Christopher Manning, Associate Professor of Computer Science and Linguistics, Stanford University ... **Chris Manning** works on systems and formalisms that can ...
nlp.stanford.edu/~manning - Cached

Δυναμικές Περιλήψεις

- Για τον υπολογισμό τους *χρειαζόμαστε τα ίδια τα έγγραφα* (δεν αρκεί το ευρετήριο)
- Cache εγγράφων – που πρέπει να ανανεώνεται
- Συχνά όχι όλο το έγγραφο αν είναι πολύ μεγάλο, αλλά κάποιο πρόθεμα του
- Βρες μικρά παράθυρα στα έγγραφα που περιέχουν όρους του ερωτήματος
  - Απαιτεί γρήγορη αναζήτηση παράθυρου στην cache των εγγράφων

Δυναμικές Περιλήψεις

- Βαθμολόγησε κάθε παράθυρο ως προς το ερώτημα
  - Με βάση διάφορα χαρακτηριστικά: το πλάτος του παραθύρου, τη θέση του στο έγγραφο, κλπ
  - Συνδύασε τα χαρακτηριστικά
- Δύσκολο να εκτιμηθεί η ποιότητα
- Positional indexes
- Ο χώρος που διατίθεται για τα παράθυρα είναι μικρός

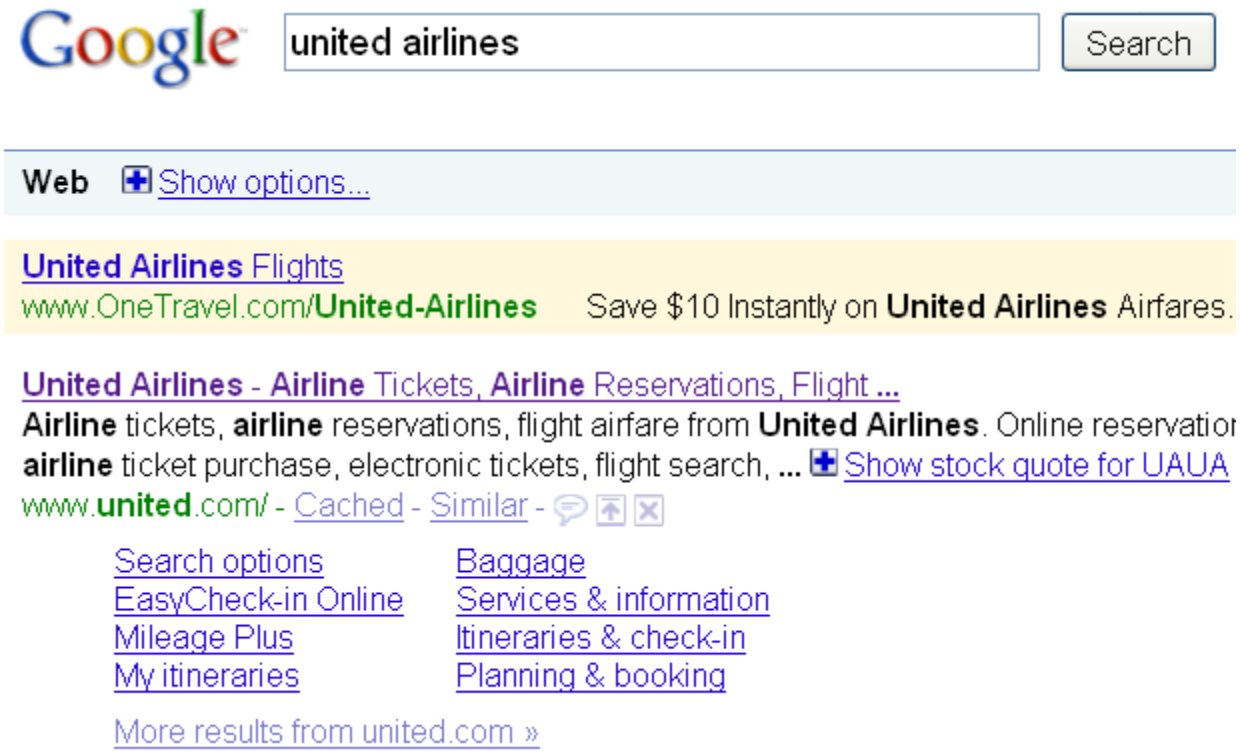
Δυναμικές Περιλήψεις

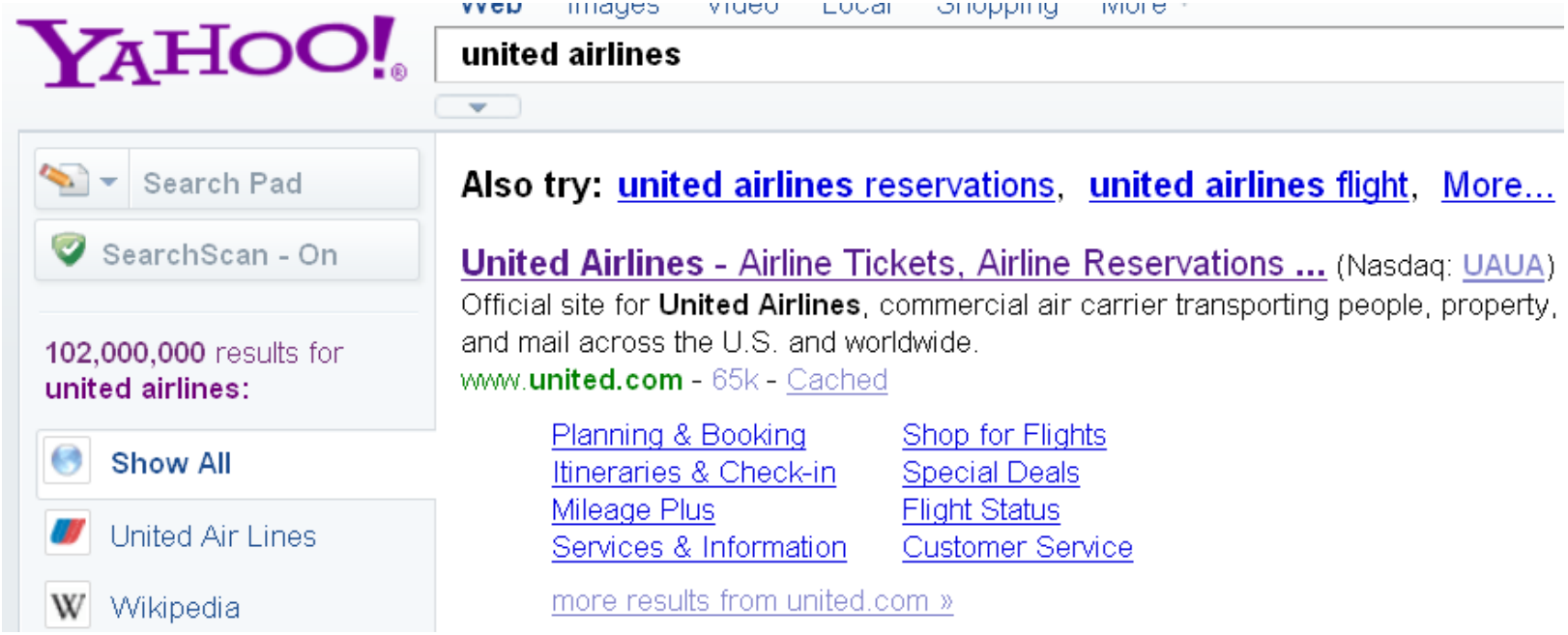
Query: "new guinea economic development" Snippets (in bold) that were extracted from a document: . . .

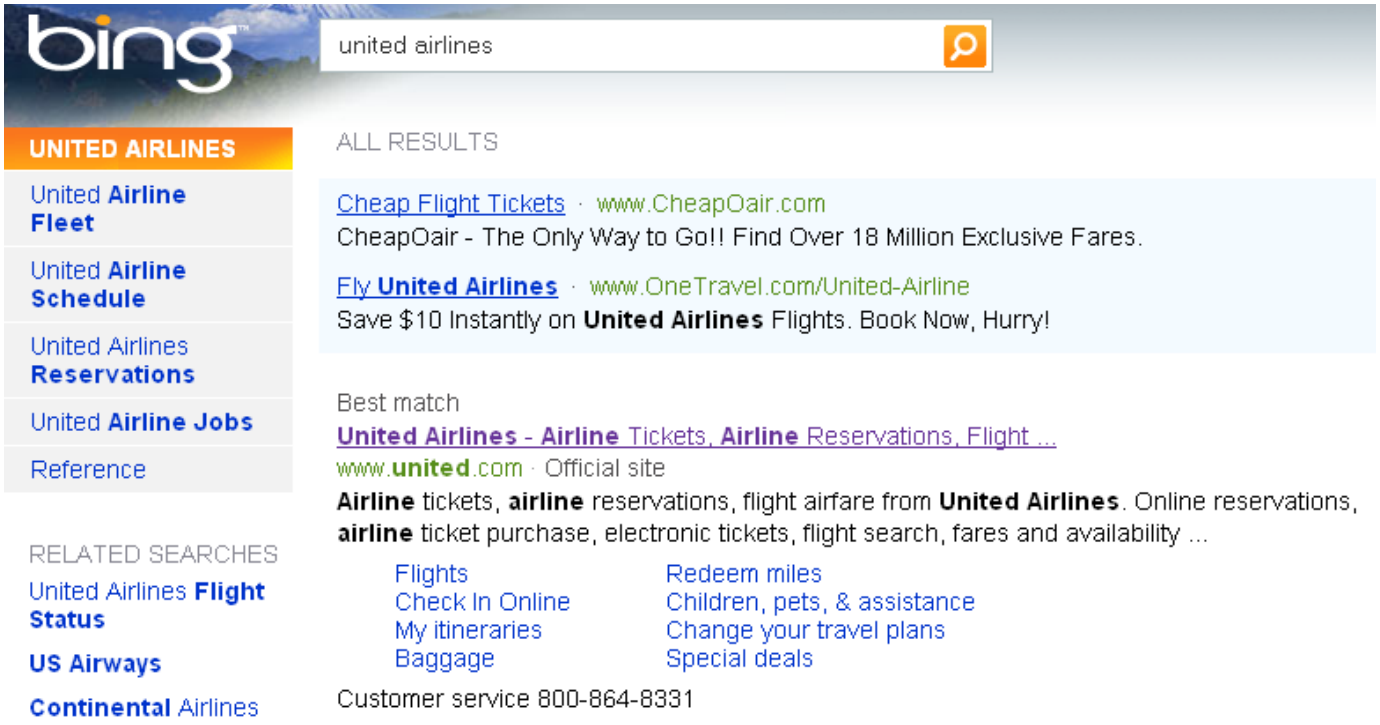
In recent years, Papua New Guinea has faced severe economic difficulties and economic growth has slowed, partly as a result of weak governance and civil war, and partly as a result of external factors such as the Bougainville civil war which led to the closure in 1989 of the Panguna mine (at that time the most important foreign exchange earner and contributor to Government finances), the Asian financial crisis, a decline in the prices of gold and copper, and a fall in the production of oil. **PNG's economic development record over the past few years is evidence that** governance issues underly many of the country's problems. Good governance, which may be defined as the transparent and accountable management of human, natural, economic and financial resources for the purposes of equitable and sustainable development, flows from proper public sector management, efficient fiscal and accounting mechanisms, and a willingness to make service delivery a priority in practice. . . .

Quicklinks

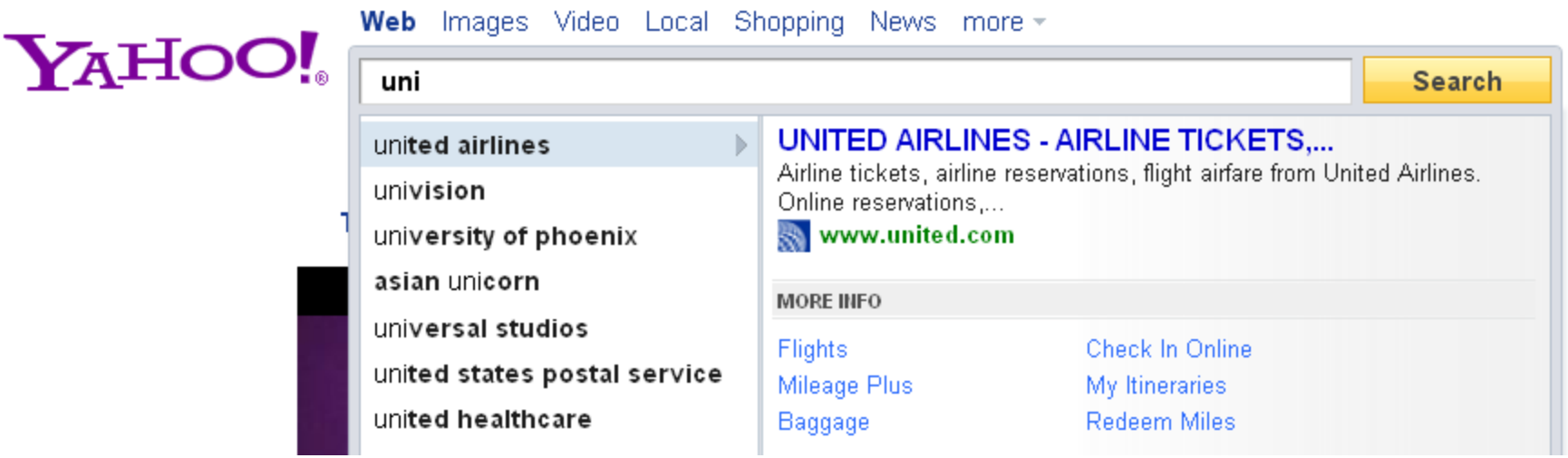
- Για *navigational query (όταν ψάχνουμε μια συγκεκριμένη σελίδα)* όπως ***united airlines*** οι χρήστες πιθανόν να ικανοποιούνται από τη σελίδα www.united.com
- Quicklinks παρέχουν navigational cues σε αυτή τη σελίδα




YAHOO!
Web Images Video Local Shopping More
united airlines
Search Pad
SearchScan - On
102,000,000 results for
united airlines:
Show All
United Air Lines
Wikipedia
Also try: united airlines reservations, united airlines flight, More...
United Airlines - Airline Tickets, Airline Reservations ... (Nasdaq: UAUA)
Official site for United Airlines, commercial air carrier transporting people, property, and mail across the U.S. and worldwide.
www.united.com - 65k - Cached
Planning & Booking
Itineraries & Check-in
Mileage Plus
Services & Information
Shop for Flights
Special Deals
Flight Status
Customer Service
more results from united.com »


bing
united airlines
UNITED AIRLINES
United Airline Fleet
United Airline Schedule
United Airlines Reservations
United Airline Jobs
Reference
RELATED SEARCHES
United Airlines Flight Status
US Airways
Continental Airlines
ALL RESULTS
Cheap Flight Tickets · www.CheapOair.com
CheapOair - The Only Way to Go!! Find Over 18 Million Exclusive Fares.
Fly United Airlines · www.OneTravel.com/United-Airline
Save $10 Instantly on United Airlines Flights. Book Now, Hurry!
Best match
United Airlines - Airline Tickets, Airline Reservations, Flight ...
www.united.com · Official site
Airline tickets, airline reservations, flight airfare from United Airlines. Online reservations, airline ticket purchase, electronic tickets, flight search, fares and availability ...
Flights
Check In Online
My itineraries
Baggage
Redeem miles
Children, pets, & assistance
Change your travel plans
Special deals
Customer service 800-864-8331

Εναλλακτικές αναπαραστάσεις;



Ερωτήσεις;