Open Source IR Tools and Libraries

Giorgos Vasiliadis, gvasil@csd.uoc.gr

CS-463 Information Retrieval Models

Computer Science Department

University of Crete

Outline

- Google Search API

- Lucene

- Terrier

- Lemur

Google Search API

Google Search API: Overview

- The API exposes the Google engine to developers.
  - You can write scripts that access the Google search in real-time.
- Google no longer issuing new API keys for the SOAP Search API.
- Instead, Google provides an AJAX Search API.
  - You can put Google Search in your web pages with JavaScript.

Google Search API: SOAP

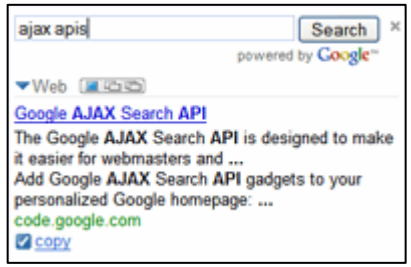
- Based on the Web Services Technology SOAP (the XML-based Simple Object Access Protocol).
- Developers write software programs that connect remotely to the Google SOAP Search API service.
- Developers can issue search requests to Google's index of billions of web pages and receive results as structured data, access information in the Google cache and check the spelling of words.
- Limitations
  - Default limit of 1,000 queries per day.
  - Can only query for 10 results a time
  - Can only access Google Web Search (not Google Images, Google Groups and so on).

Google Search API: AJAX

- Lets you put Google Search in your web pages with JavaScript.
- Does not have a limit on the number of queries per day.
- Supports additional features like Video, News, Maps, and Blog search results.

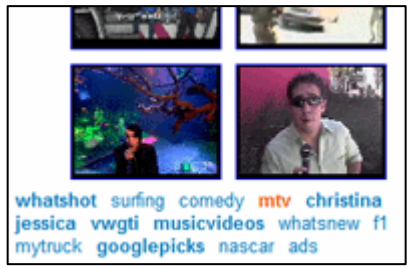
Google Search API: AJAX

- Web Search
  - Incorporate results from Web Search, News Search, and Blog Search

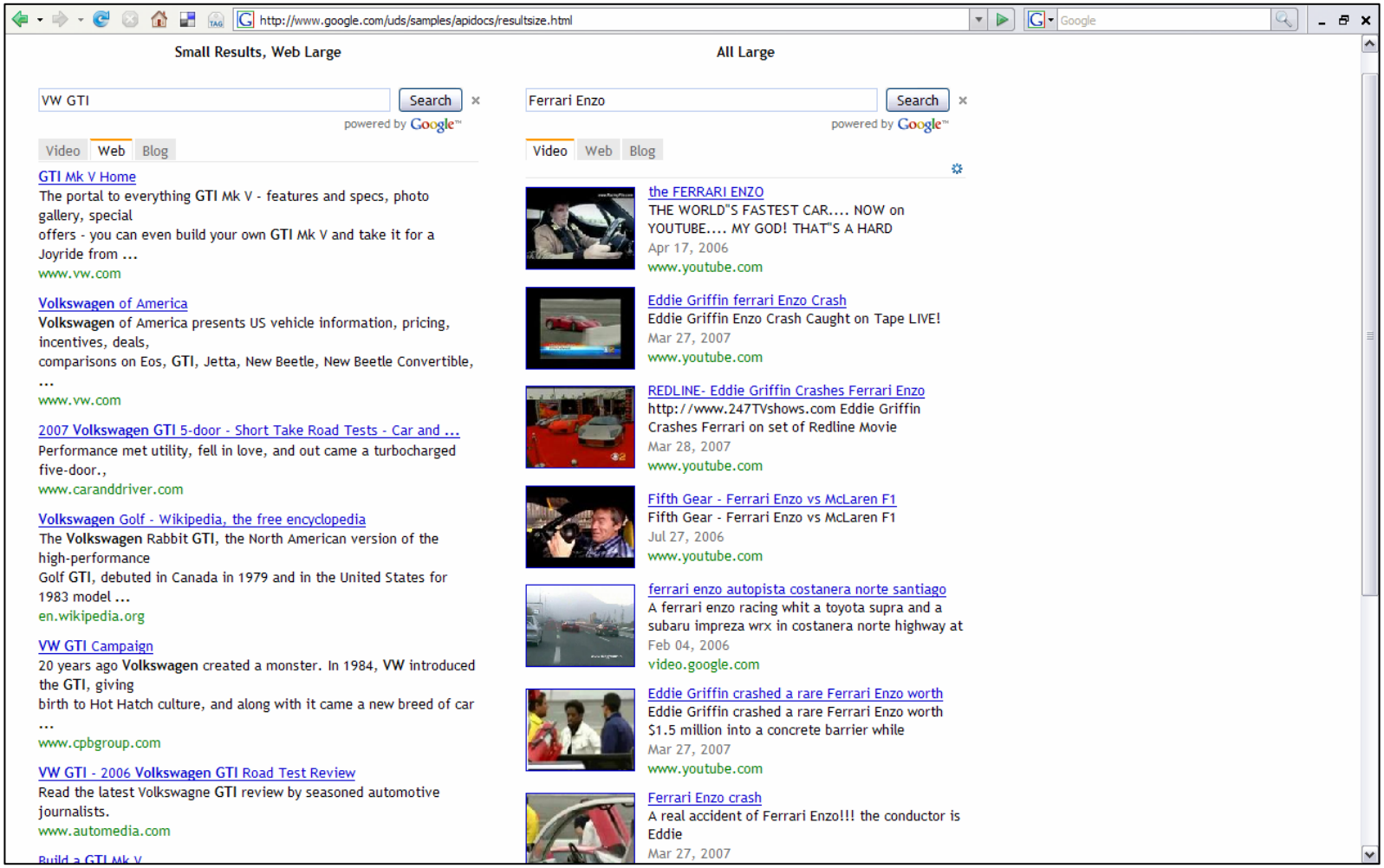


- Local Search
  - Provides access to local search results from Google Maps.

- Video Search
  - Incorporate a simple search box
  - incorporate dynamic, search powered strips of video and book thumbnails.



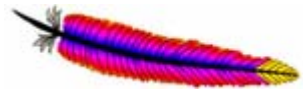
Google Search API: Demo



Google Search API: References

- Google SOAP Search API
  http://code.google.com/apis/soapsearch/
- Google AJAX Search API
  http://code.google.com/apis/ajaxsearch/
- Google AJAX Search API Developer Guide
  http://code.google.com/apis/ajaxsearch/documentation/
- Google AJAX Search API Samples
  http://code.google.com/apis/ajaxsearch/samples.html

Lucene



Lucene

- Doug Cutting's grandmother's middle name
- Cross-Platform API
- Implemented in Java
  - Ported in C++, C#, Perl, Python
- Offers scalable, high-performance indexing
  - Incremental indexing as fast as bath indexing
  - Index size roughly 20-30% the size of indexed text
- Supports many powerful query types

Lucene: Modules

- Analysis
  - Tokenization, Stop words, Stemming, etc.
- Document
  - Unique ID for each document
  - Title of document, date modified, content, etc.
- Index
  - Provides access and maintains indexes.
- Query Parser
- Search / Search Spans

Lucene: Indexing

- A `Document` is a collection of `Fields`

- A `Field` is free text, keywords, dates, etc.
- A `Field` can have several characteristics
  - indexed, tokenized, stored, term vectors
  - Apply `Analyzer` to alter `Tokens` during indexing
    - Stemming
    - Stop-word removal
    - Phrase identification

Lucene: Searching

- Uses a modified Vector Space Model
- We convert a user's query into an internal representation that can be searched against the Index
- Queries are usually analyzed in the same manner as the index
- Get back `Hits` and use in an application

Lucene: Query Parser Syntax

- Terms
  - Single terms and phrases
- Fields
  - E.g. title:"Do it right" AND right
- Wildcard Searches
  - '?' for single character
  - '*' for multiple characters
- Proximity Searches
  - "jakarta apache"~10
- Fuzzy Searches
  - Levenshtein Distance or Edit Distance algorithm
- Range Searches
  - mod_date:[20020101 TO 20030101]
  - title:{Aida TO Carmen}
- Boosting a Term
  - E.g. jakarta^4 apache
- Boolean Operators

Lucene: More Advanced Options

- Relevance Feedback
  - Manual
    - User selects which documents are relevant/non-relevant
    - Get the terms from the term vector for each of the documents and construct a new query.
  - Automatic
    - Application assumes the top X documents are relevant and the bottom Y are non-relevant and constructs a new query based on the terms in those documents.
- Span Queries
- Phrase Matching

Lucene: Basic Demo

- The latest version can be obtained from http://www.apache.org/dyn/closer.cgi/lucene/java/
- To build an index just type
  - `java org.apache.lucene.demo.IndexFiles <dir>`
- To search from an index type:
  - `java org.apache.lucene.demo.SearchFiles <index>`


Terrier


EPSRC
Engineering and Physical Sciences
Research Council

Terrier

Terrier: Overview

- Stands for **TER**abyte **R**etr**IE**ve**R**.
- Open Source API (Mozilla Public Licence).
- Written in cross-platform Java.
- Highly compressed disk data structures.
- Handling large-scale document collections.
- Standard evaluation of TREC ad-hoc and known-item search retrieval results.

Terrier: Indexing

- Create your own `Collection` decoder and `Document` implementation.
  - Centralized or distributed Setting.
- Indexer iterates through the collection and creates the following data structures
  - Direct Index
  - Document Index
  - Lexicon

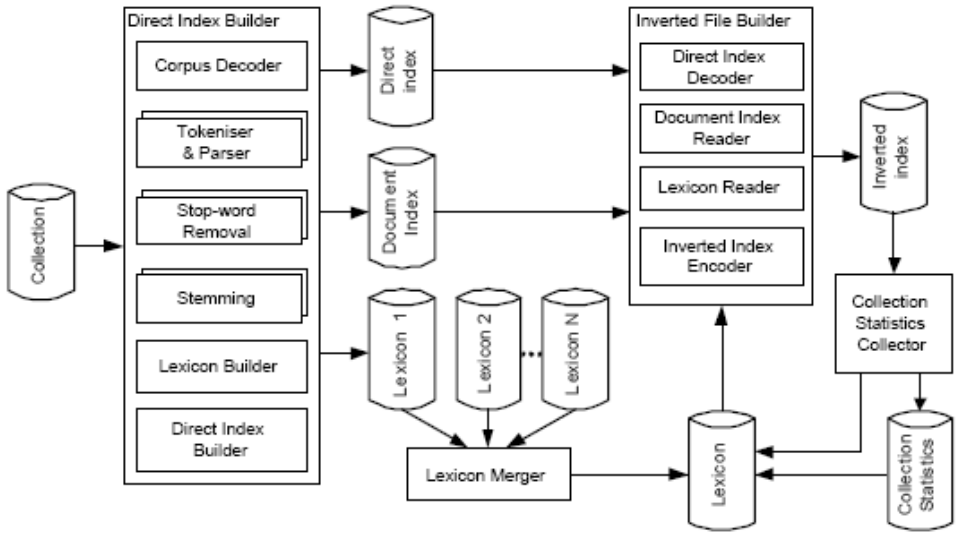


Fig. 1. Indexing process with Terrier.

Terrier: Indexing

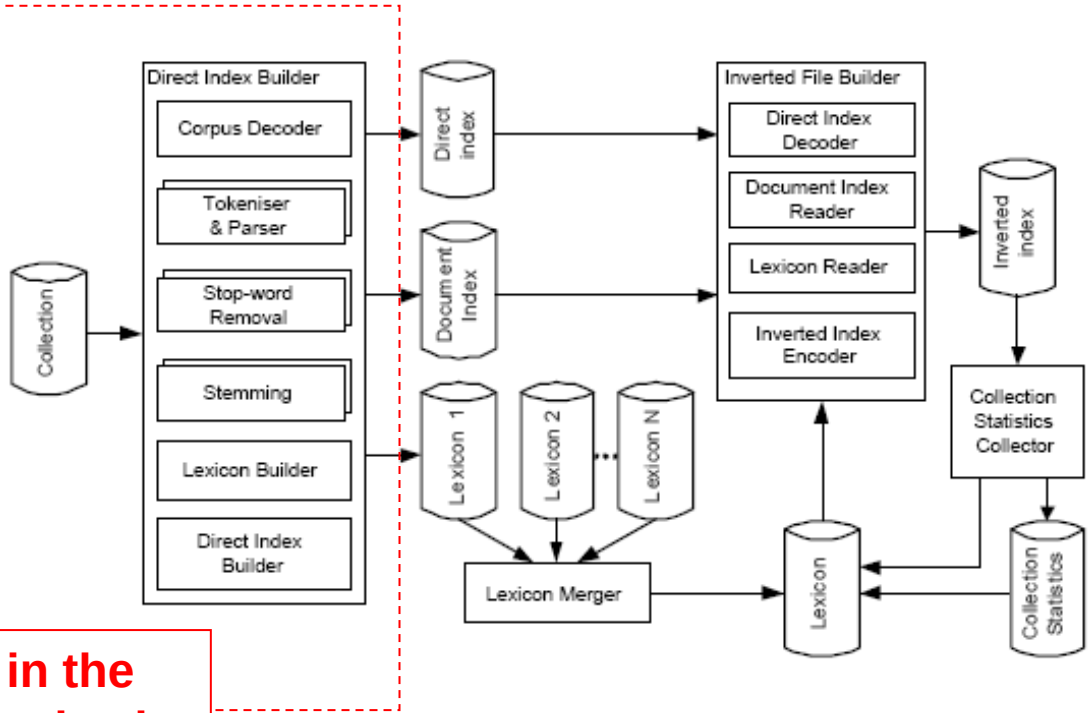


Fig. 1. Indexing process with Terrier.

Each document in the collection is tokenized and parsed.

Terrier: Indexing

In this way, we build the direct and document indices.

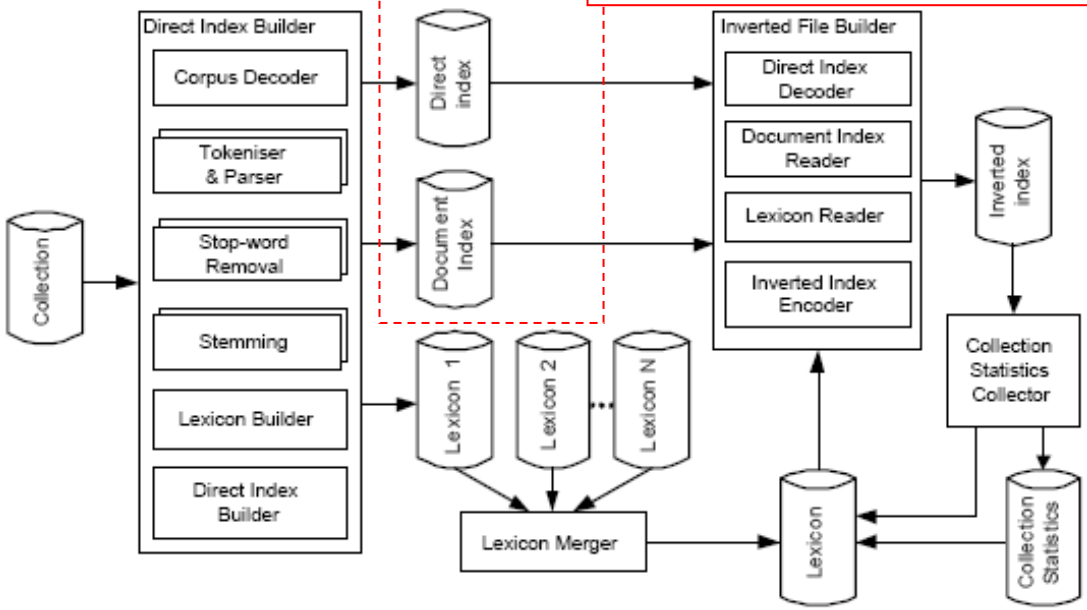


Fig. 1. Indexing process with Terrier.

Terrier: Indexing

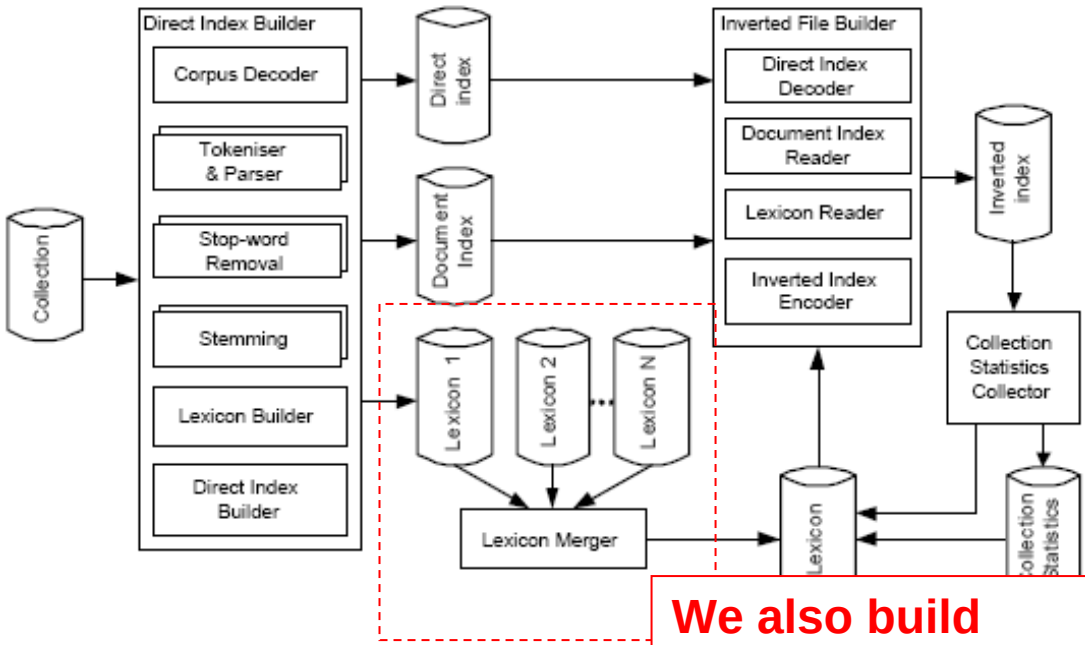


Fig. 1. Indexing process

We also build temporary lexicons in order to reduce the required memory during indexing

Terrier: Indexing

The inverted index is built from the existing direct index, document index and lexicon

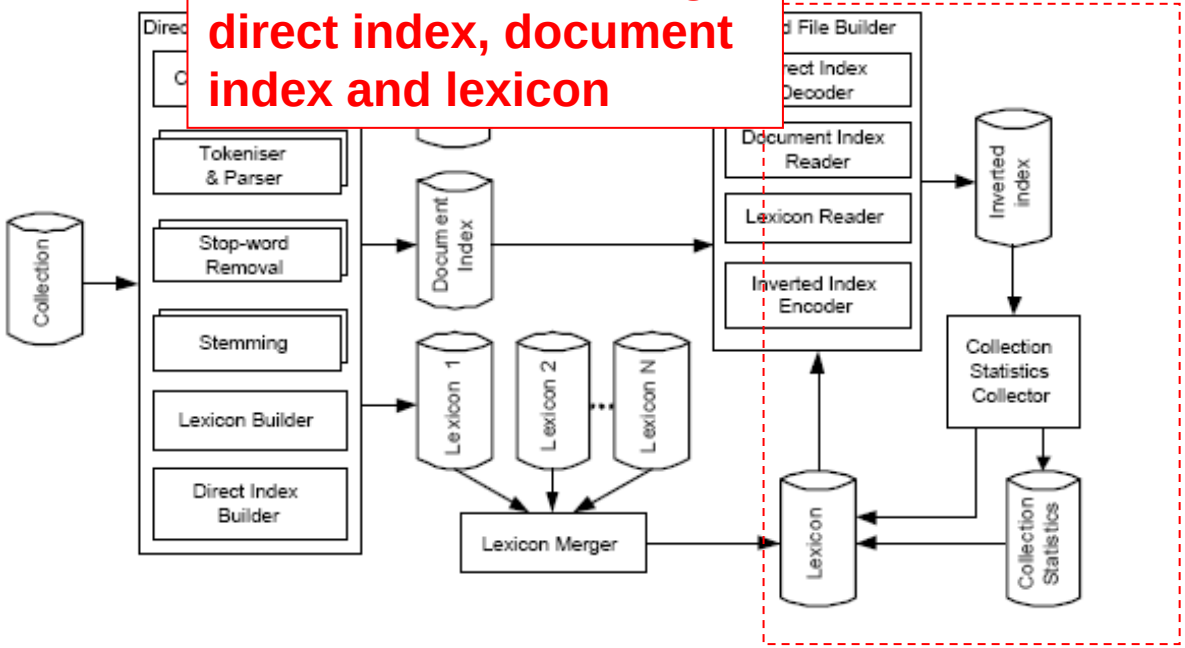


Fig. 1. Indexing process with Terrier.

Terrier: Retrieval

- Parsing
- Pre-processing
- Matching
- Post Processing
- Post Filtering

- Query Language
  - term1 term2
  - term1^2.3
  - +term1 -term2
  - "term1 term2"~n

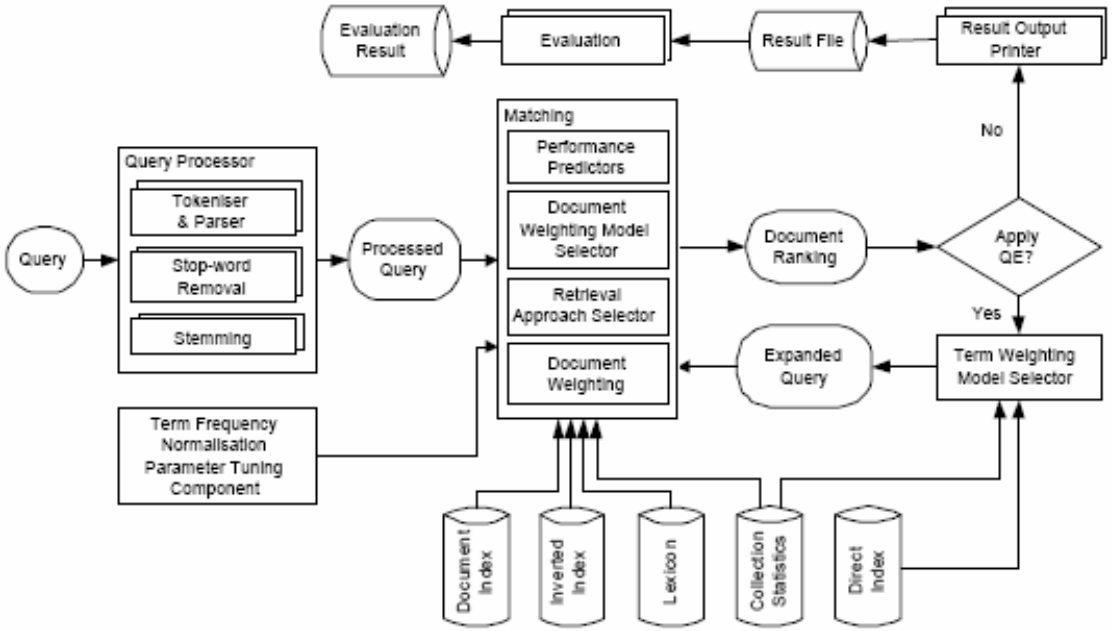


Fig. 2. Retrieval process with Terrier.

Terrier: Retrieval

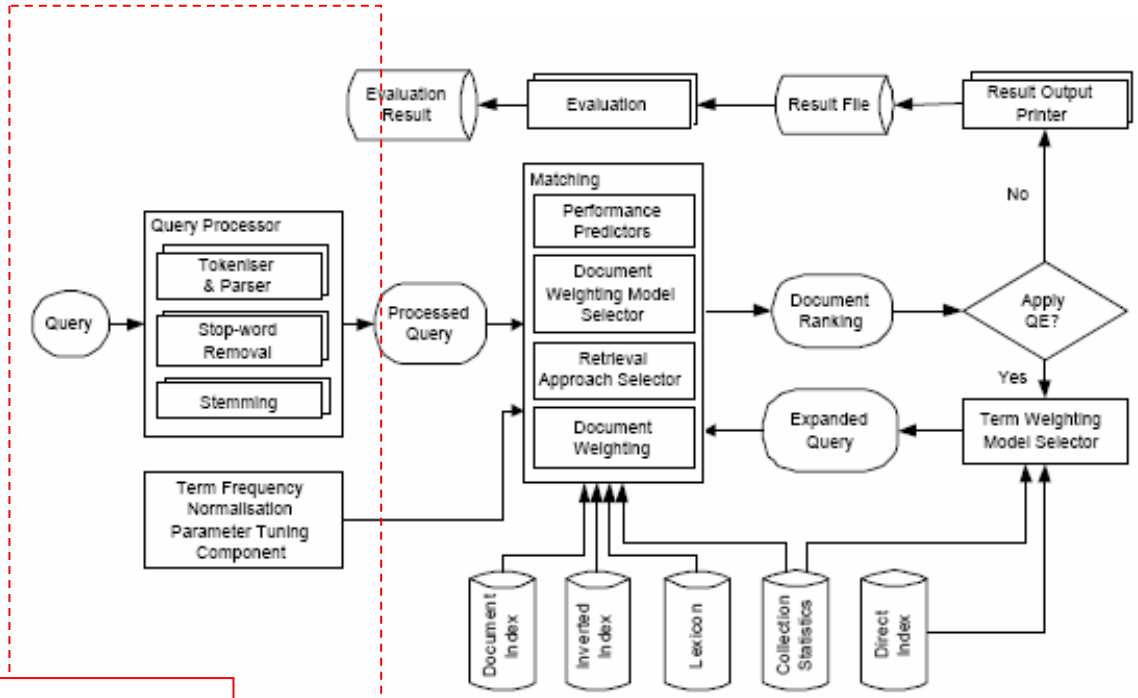


Fig. 2. Retrieval process with Terrier.

Remove stop words and apply stemming to the query.

Terrier: Retrieval

Terrier automatically select the optimal document weighting model

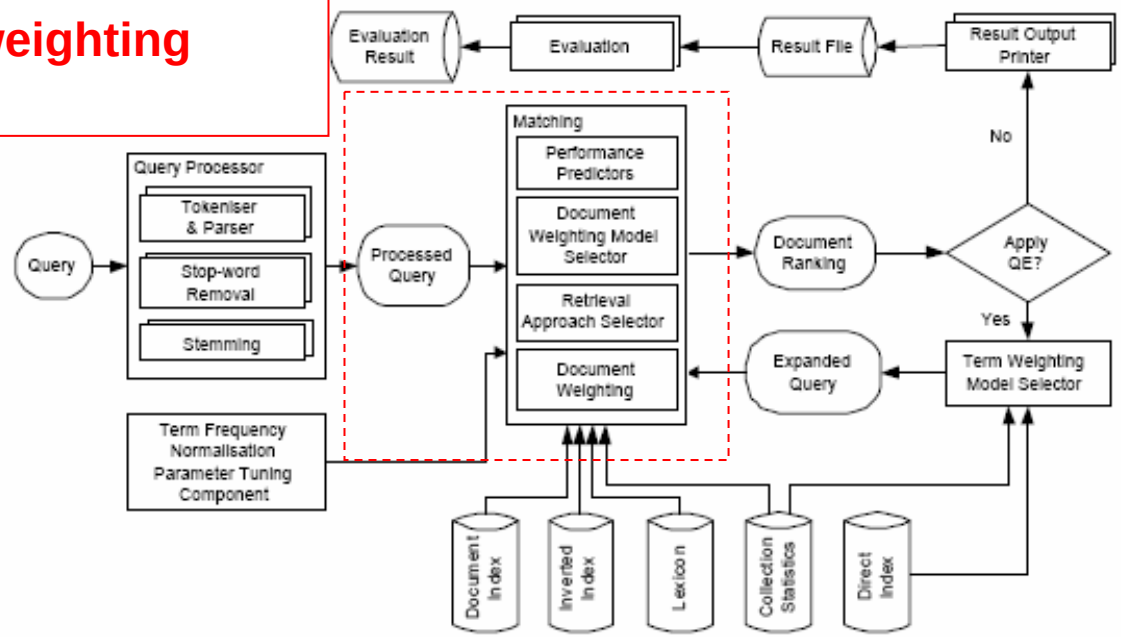


Fig. 2. Retrieval process with Terrier.

Terrier: Retrieval

If Query Expansion is applied an appropriate term weighting model is selected and the most informative terms from the top ranked documents are added to the query

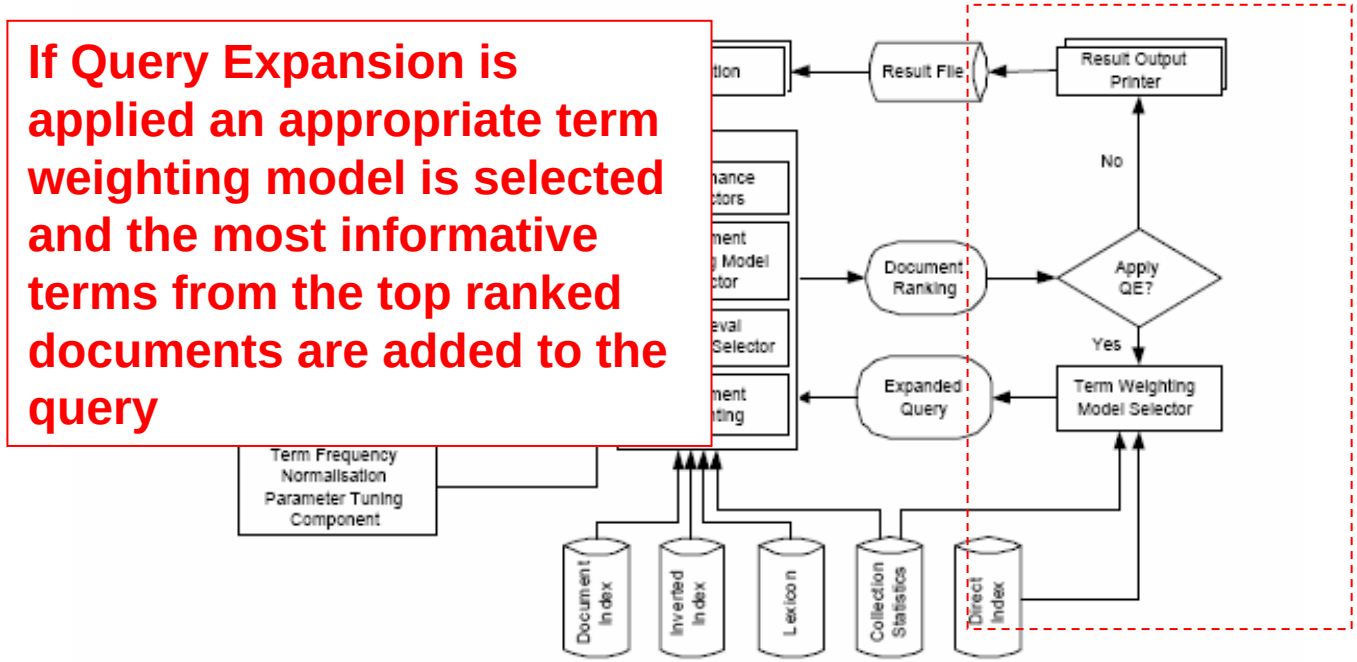


Fig. 2. Retrieval process with Terrier.

Terrier: Sample Applications

- Trec Terrier
  - An application that allows Terrier to index and retrieve from standard TREC test collections.
  - Instructions are available at http://ir.dcs.gla.ac.uk/terrier/doc/trec_terrier.html

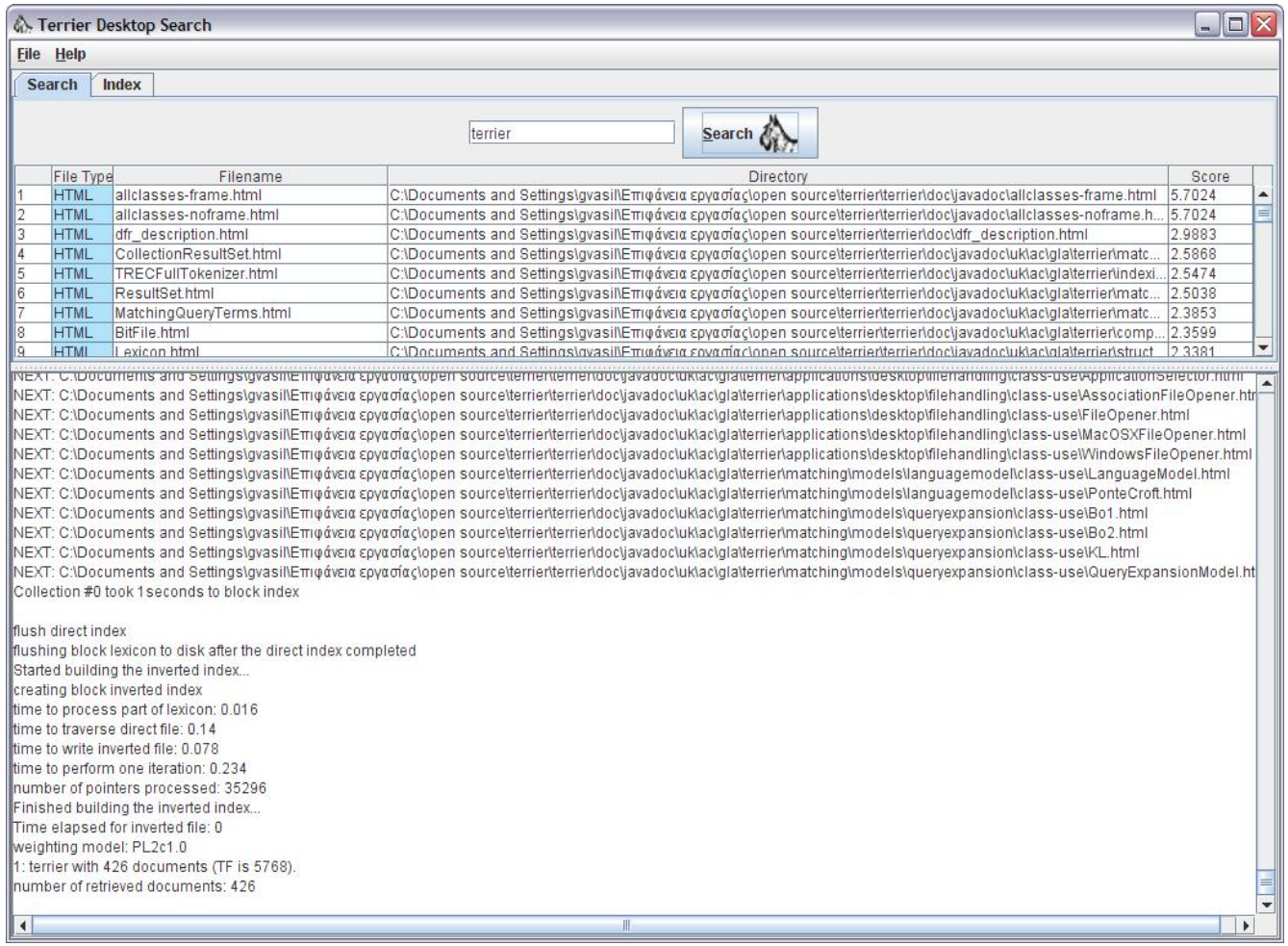
Terrier: Sample Applications

- Desktop Search
  - A Swing (graphical) application that can be used to index files from the local machine, and then perform queries on them.
  - The scripts for running the desktop search application are:
    - desktop_search.sh (Linux, Mac OSX)
    - desktop_search.bat (Windows)

Terrier: Sample Applications

- Interactive Querying
  - A console application for performing simple queries on an existing index and seeing which documents are returned.
  - The scripts for running the console application are:
    - interactive_terrier.sh (Linux, Mac OS X)
    - interactive_terrier.bat (Windows)

Terrier: Demo



Lemur

Lemur: Overview

- Support for XML and structured document retrieval
- Interactive interfaces for Windows, Linux, and Web
- Cross-Platform, fast and modular code written in C++
- C++, Java and C# APIs
- Free and open-source software

Lemur: API

- Provides interfaces to Lemur classes that are grouped at three different levels:
  - Utility level
    - Common utilities, such as memory management, document parsing, etc.
  - Indexer level
    - Converts a raw text collection to data structures for efficient retrieval.
  - Retrieval level
    - Abstract classes for a general retrieval architecture and concrete classes for several specific information retrieval

Lemur: Indexing

- Multiple indexing methods for small, medium and large-scale (terabyte) collections.
- Built-in support for English, Chinese and Arabic text.
- Porter and Krovetz word stemming.
- Incremental indexing.

Lemur: Retrieval

- Supports major language modelling approaches such as Indri and KL-divergence, as well as vector space, tf-idf, Ocapi and InQuery
- Relevance- and pseudo-relevance feedback
- Wildcard term expansion (using Indri)
- Supports arbitrary document priors (e.g., Page Rank, URL depth)

Questions ?


"That's all Folks!"
Cartoon Songs From
MERRIE MELODIES & LOONEY TUNES