DSC650: Data Technology and Future Emergence
Lecture 6 : Searching & Indexing Big Data

Dr. Khairul Anwar Sedek
Faculty of Computer and Mathematical Sciences
Universiti Teknologi MARA
Perlis Branch

- Lecture 6: **Searching & Indexing Big Da**

  Full Text Indexing and Searching
  Indexing with Lucene
  Distributed Searching with Elastic Search

At the end of the lecture, students should be able to;

- CLO1: Demonstrate an understanding on the basic **concepts and practices** of big data technology

Source:
Shahi, D. (2015) Apache Solr: A Practical Approach to Enterprise Search. Apress.

Search

Search is ubiquitous and has become part of our digital life!

Open an e-commerce web site, and the search box appears right at the center or top of the page.

Basically, every portal that has data to expose needs a search engine.

How Search Works

Search Engines

Search engines are one of the most widely used implementations of information retrieval systems.

Search Engines categories

Web search:

- This **crawls** through the World Wide Web, **extracts** the contents from HTML tags, and **indexes** the information to make it searchable.
- Some engines also crawl for images and other associated files, and before indexing process the text for other interesting information and links.
- DuckDuckGo is a typical example of a web search engine developed using Solr.

Vertical search:

- Vertical search addresses a **specific domain**, an area of information or industry such as healthcare or finance.
- The information is generally available in a primary database or data source such as a Word document or PDF file.

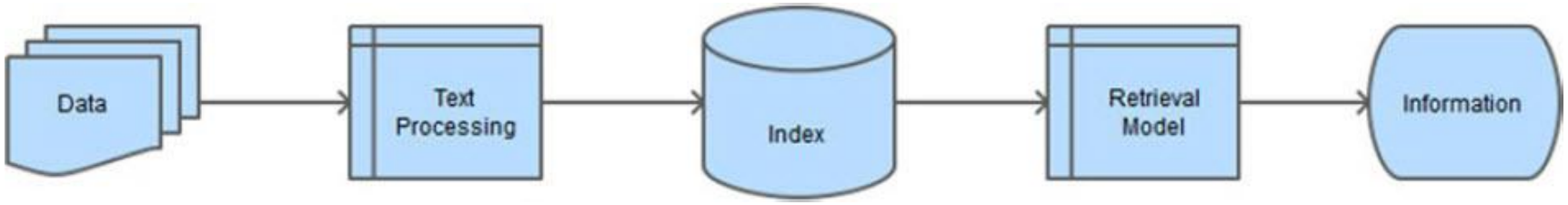
Desktop search:

- This is designed to **index the files in a user's PC** and make it searchable. Some search engines also index the file content along with the metadata to make it searchable. Spotlight in Mac OS X is a typical example.

Others:

- Information retrieval is no longer limited to text and is being widely used for searching image, audio fingerprints, and in speech recognition.

Search Engines

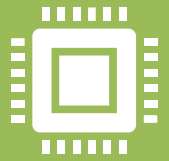


A search system

- The goal of any information retrieval system is to turn data into actionable information that satisfies the users' needs.
- In case of search engines, information pertains to relevant documents that are retrieved for search requests.

Solr and Lucene

Solr (pronounced *solar)*, is an enterprise-ready, blazingly fast, and highly scalable search platform built using Apache Lucene.

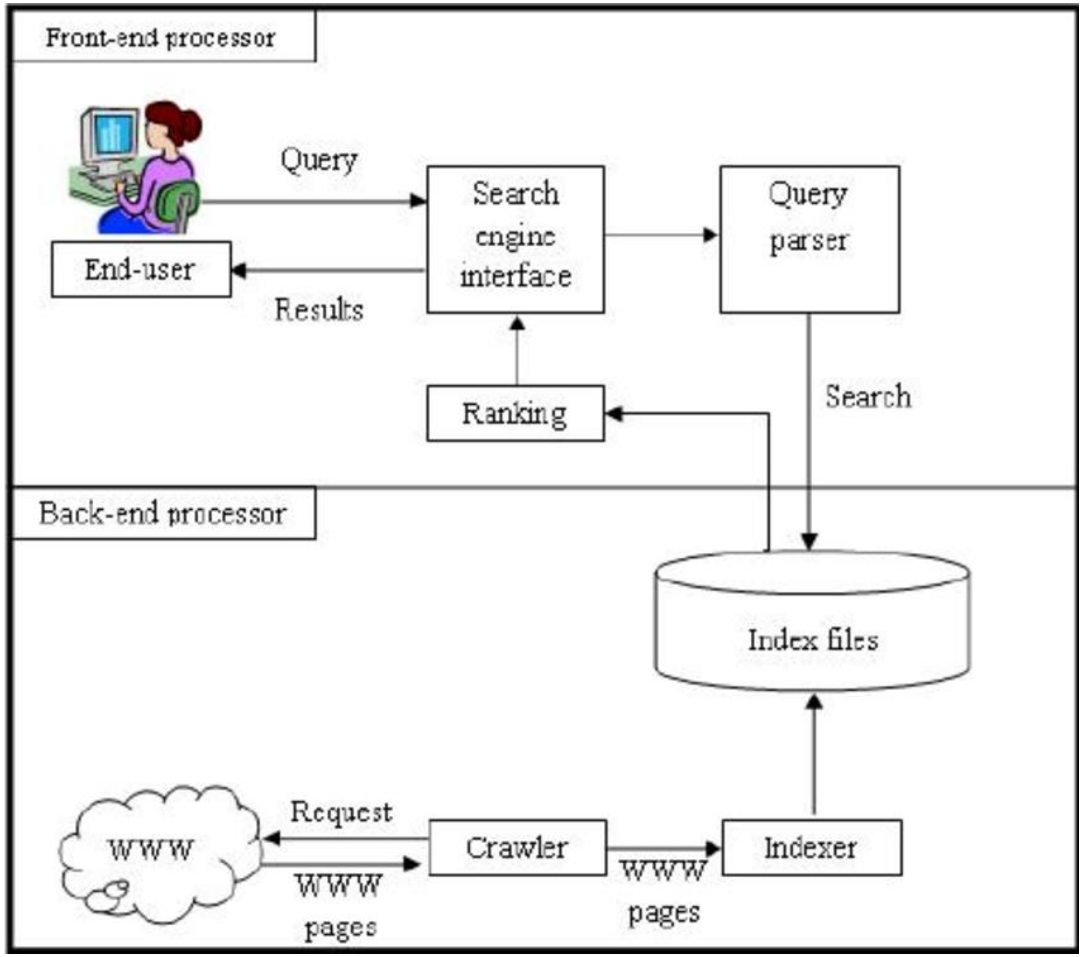


Solr is written in Java and runs as a stand-alone server.

Apache Lucene is a popular open-source search library written entirely in Java.

It is widely used for indexing a large collection of documents and supporting full-text search.

Architecture and main components of standard search engine model.



https://www.researchgate.net/publication/220717128_A_web_search_engine_model_based_on_index-query_bit-level_compression

History - Solr and Lucene

Solr 1.x: **This was an enterprise-ready version of the product.**

Solr 1.3 was released in January 2007. Solr 1.4 was released in November 2009 with enhancements to indexing, searching, replication, and rich document indexing and better database integration.

Solr 3.x: **In March 2011, development of Solr and Lucene projects merged, so there was no 2.x version of Solr.**

The next release of Solr was directly labeled 3.1 to match the Lucene version. Solr 3.x was focused on feature additions such as spatial search, and integrations such as UIMA for processing pipelines, Velocity for the UI, and so on.

Solr 4.x: **Solr 4.0 was primarily about distributed search and about SolrCloud making Solr even more reliable, fault-tolerant, and scalable.**

including the new SolrCloud feature

Solr 5.x: **Solr 5.0 was released in February 2015 with a focus on ease of use and hardening.**

Recent releases have focused on security, analytics and extending the APIs. Refer to the next section for details.

Solr 7.0 released in Sept 2017

dded support multiple replica types, auto-scaling, and a Math engine.

Most recent Apache Solr 8.5.1 (May 2020)

In March 2019

Solr nodes can now listen and serve HTTP/2 requests

Lucene vs Solr

- A simple way to conceptualize the relationship between Solr and Lucene:
  - **Solr** is that of a **car**
  - and its **engine** is **Lucence**.
- You can't drive an engine, but you can drive a car.
- Similarly, Lucene is a programmatic library which you can't use as-is, whereas Solr is a complete application which you can use out-of-box.

Introducing Apache Solr

Run Your Own Search Engine with:

Solr

-Beginner Friendly -Docker -Node App

Major Building Block/Components

Request Handler:

All the requests that you make to Solr are processed by one of the classes implementing SolrRequestHandler.

You configure the handler to map to a specific URI endpoint, and the requests made to this endpoint start getting served by it.

Search Component:

A search component defines the logic to implement the features provided by the search handler.

These components should be registered in a SearchHandler, a request handler which serves the user query.

For example, features such as query, spell-checking, faceting, and hit-highlighting are implemented as components and are registered to SearchHandler.

Multiple components can be registered to a search handler.

Query Parser:

This translates the user query into instructions that Lucene understands.

They are generally registered in the SearchComponent, a component that defines the logic for performing a search.

Lucence Major Building Block/Components

Similarity:

This class determines how Lucene weights terms and scores a document.

If you want to change the scoring behavior, this is the class to extend.

Response Writer:

This decides how the response to the user query should be formatted.

For each response type such as XML, JSON, or Velocity, there exists a separate response writer.

Analyzer/tokenizer:

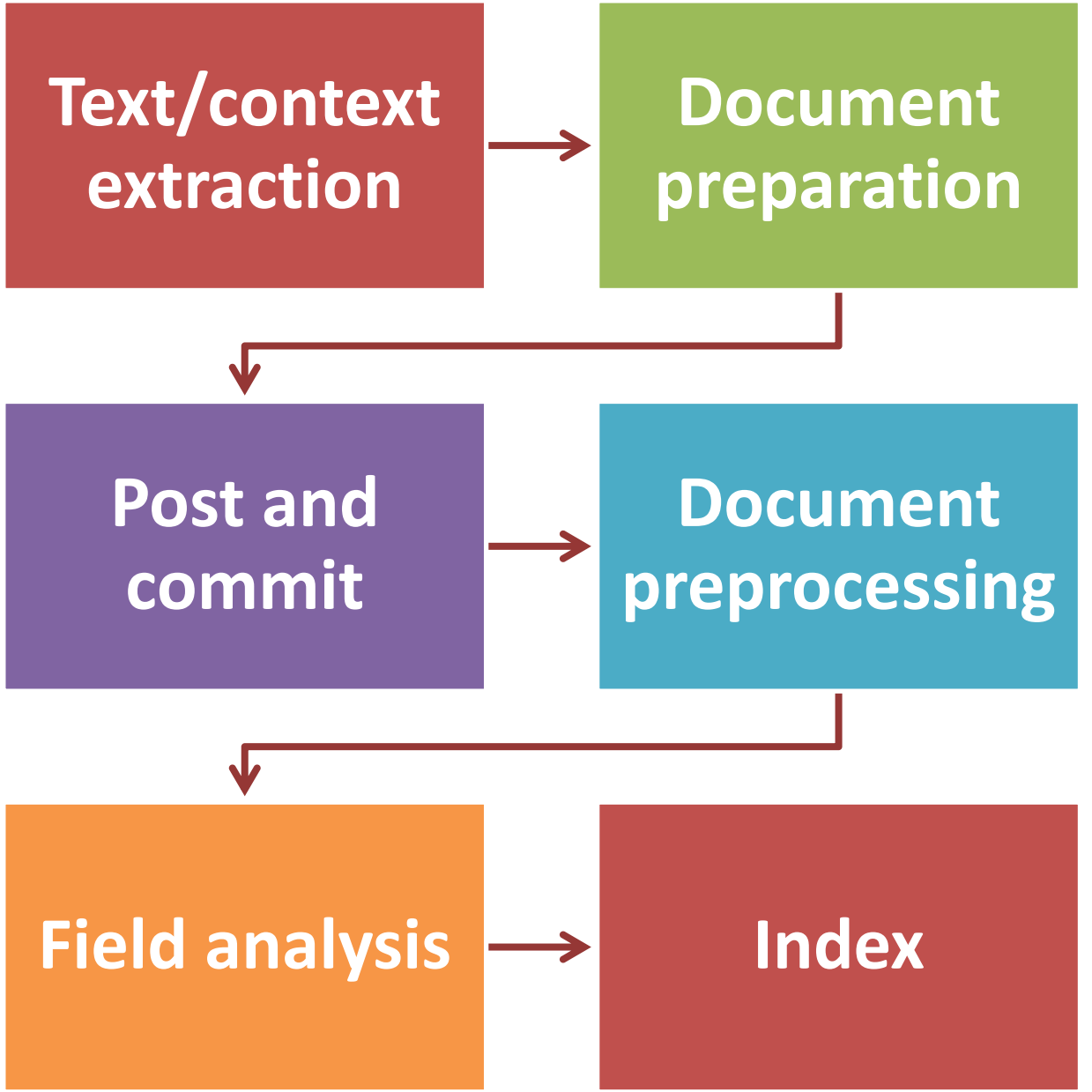
The smallest unit of data that Lucene understands is a token.

The implementations of Analyzer, tokenizer or TokenFilter decide how to break the text into tokens.

Update Request Processor:

While indexing your document, you can invoke a set of UpdateRequestProcessor as part of your chain to perform custom actions upon your data.

Information Retrieval Component & Process



Information Retrieval Component & Process: **Solr Indexing Process**

- **Text extraction:**
  - to **extract the text for indexing**.
  - The text can be acquired, for example, by reading files, querying databases, crawling web pages, or reading RSS feeds. Extraction can be performed by your Java client application or Solr components.
- **Document preparation:**
  - The extracted text should be **transformed into a Solr document** for ingestion.
  - The prepared document should adhere to the native format specified, for example, for XML or JSON.
- **Post and commit:**
  - During this process, you **post the document to the appropriate Solr endpoint** with required parameters.
  - Solr - provided extraction capabilities are performed based on the endpoint you invoke.
- **Document pre-processing:**
  - You might want to do **clean-up, enrichment, or validation of text** received by Solr.
- **Field analysis:**
  - Field analysis **converts the input stream into terms**.
  - This step refers to the analysis chain of analysers, tokenizers and token filters that are applied on the fieldType definition, which you read about in the previous chapter.
- **Index:**
  - The terms output from **field analysis are finally indexed**; the inverted index is created.
  - These indexed terms are **used for matching and ranking in search requests**.
  - After you trigger the post operation (mentioned in Step 3), the pre-processing and field analysis defined in Solr will be triggered automatically and documents will be indexed.

Information Retrieval Component & Process: ***Context extraction***

- is the first process in information flow.
- In this process, you **extract the content from the data source and convert it into indexable documents**.
- In Solr, the process can be performed inside it or in the client application that indexes the documents.

File Formats	Open Source Tools
PDF	Apache POI Apache PDFBox
Microsoft Word/Excel/PowerPoint	Apache POI Apache OpenOffice
HTML	jsoup HTMLCleaner
XML	Apache Xerces Java Architecture for XML Binding (JAXB) Dom4j Many other SAX and DOM parsers are available Solr DataImportHandler
OCR	Tesseract
Geospatial	Geospatial Data Abstraction Library (GDAL)
E-mail	JavaMail API
Web crawling	Apache Nutch

Information Retrieval Component & Process: **Text Processing**

- Raw documents received from the **database or text extracted** from binary documents, require processing before being indexed.
- The processing tasks can be **for cleansing, normalization, enrichment, or aggregation of text**.
  - The processes are generally chained together to achieve the desired output, and the pipeline depends on your processing needs, which varies depending on the nature and quality of data and the desired output.
- In Solr**, text processing** can be performed **at two steps, by the *analysis* process and *update request processors***.
  - The purpose of these steps is to address various text-processing needs.
- The **analysis process** caters to **individual field-level tokenization and analysis**,
- **update request processor** is for other text-processing needs and is scoped to the entire document.
  - **Update request processors** are also called *preprocessors* or *document processors*.

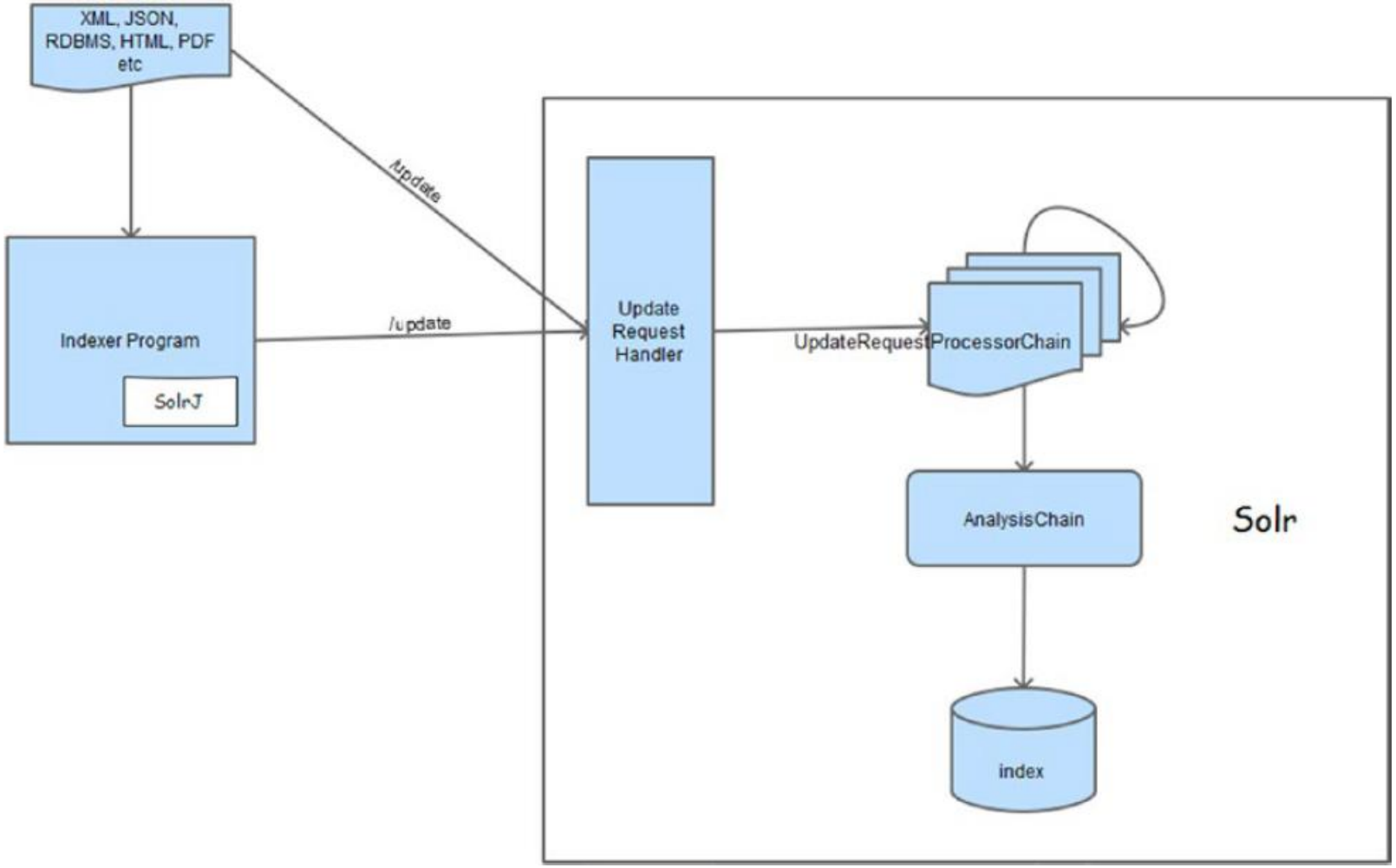
Information Retrieval Component & Process: **Text Processing**

- **Cleansing and Normalization**
  - Not all data is important, and some important data needs normalization and transformation before being indexed.
  - Common *cleansing and normalization* tasks include removal of punctuation marks, removal of stop words (*a*, *the*), lowercasing, conversion to the nearest ASCII character, and stemming.

Information Retrieval Component & Process: **Text Processing**

- ***Text enrichment***
  - ***Text enrichment*** is the phase of **analysing and mining of content to find patterns for enrichment, annotation, entity extraction, and applying a variety of intelligence** to make the content more usable, understandable, and relevant
  - *Entity extraction:* If the content being indexed is unstructured, or any of the fields is full-text, you may want to extract entities such as person, organization, and location and annotate the content.
  - *Part-of-speech tagging:* Words in any language can be classified as a predefined part of speech
  - *Thesauri Ontologies:* An ontology is a formal representation of concepts and its relationships

Information Retrieval Component & Process: **Solr Indexing Process**



Information Retrieval Component & Process: **Inverted Index**

- The data extracted from your data source should be **indexed for fast and accurate retrieval**.
- Solr internally uses Lucene, and when you add documents, it creates an inverted index to make the information searchable.
- An *inverted index* is the primary data structure in a typical search engine, which maintains a dictionary of all the unique terms and mapping to all the documents in which they appear.
- Each term is a key, and its value is a postings list, which is the list of documents the term appears in.
- An inverted index is similar to the index at the end of a book, which contains a list of words and corresponding pages in which the words occur.

Information Retrieval Component & Process: **Retrieval Models**

- **Retrieval Models**
  - To search relevant documents, you need a retrieval model.
  - A *retrieval model* is a framework for defining the retrieval process by using mathematical concepts.
    - Boolean Model
    - Vector Space Model
    - Probabilistic Model
    - Language Model
  - It provides a blueprint for determining documents relevant to a user need, along with reasoning of why a document ranks above another.

ElasticSearch

- Elasticsearch is a popular and widely used open source search engine since its release 2010.
- Both Solr and Elasticsearch are built on top of Lucene
- Solr aces Elasticsearch in terms of Lucene - development of Solr and Lucene are merged, Solr always has the same version as Lucene.
- Solr is more focused on text search and supporting features.
- But when it comes to analytics and monitoring, Elasticsearch excels.

What is Elasticsearch?

Related Technologies

Apache ZooKeeper:

This controls the heartbeat of SolrCloud. It is a **central service for maintaining cluster configurations, store statuses, and synchronization information** in a fault-tolerant and highly available manner. For SolrCloud to function, at least one ZooKeeper instance should be up and running.

Apache OpenNLP:

This is a Java library for **natural language processing of text**. It supports tasks to understand and interpret text written in human languages.

Apache UIMA:

If your data is unstructured, the Solr UIMA contrib module can be used **to give structure to the data**. UIMA allows us to **define a custom pipeline for analyzing a large volume of unstructured text and annotating the extracted metadata**, which can be added to Solr fields.

Carrot clustering:

You can **cluster your similar or semantically related search results together** by using Carrot2, and you can do so with just a few changes in your XML configuration.

Apache Tika:

This toolkit provides a **framework for parsing and extracting contents and metadata from files in many different formats** such as Word, PPT and PDF. Solr provides a Solr Cell framework which uses this toolkit for indexing contents and metadata from such files.