CSE 435/535 : Information Retrieval

Getting Started with Solr

Agenda

- Background : What is Solr? Why Solr?
- Solr installation : Lingering issues
- Quickstart : Index tweets in Schemaless mode
- Anatomy of a Solr schema
- Making some modifications
- Tips, tricks and resources

Background : Lucene / Solr

- Lucene : Basic text search engine
  - Written 100% in Java, originally written by Doug Cutting in 1999, part of the Apache Software Foundation
  - (Still) used by AOL, Apple, CiteSeerX, Eclipse, IBM, LinkedIn, Twitter etc.
    - Spawned several projects : Nutch (Web crawling + html parsing) and Solr
- Solr : Web application built using Lucene
  - Uses Lucene for most features
  - Supports several popular web servers, comes pre-packaged with a Jetty server
  - Provides distributed IR capabilities

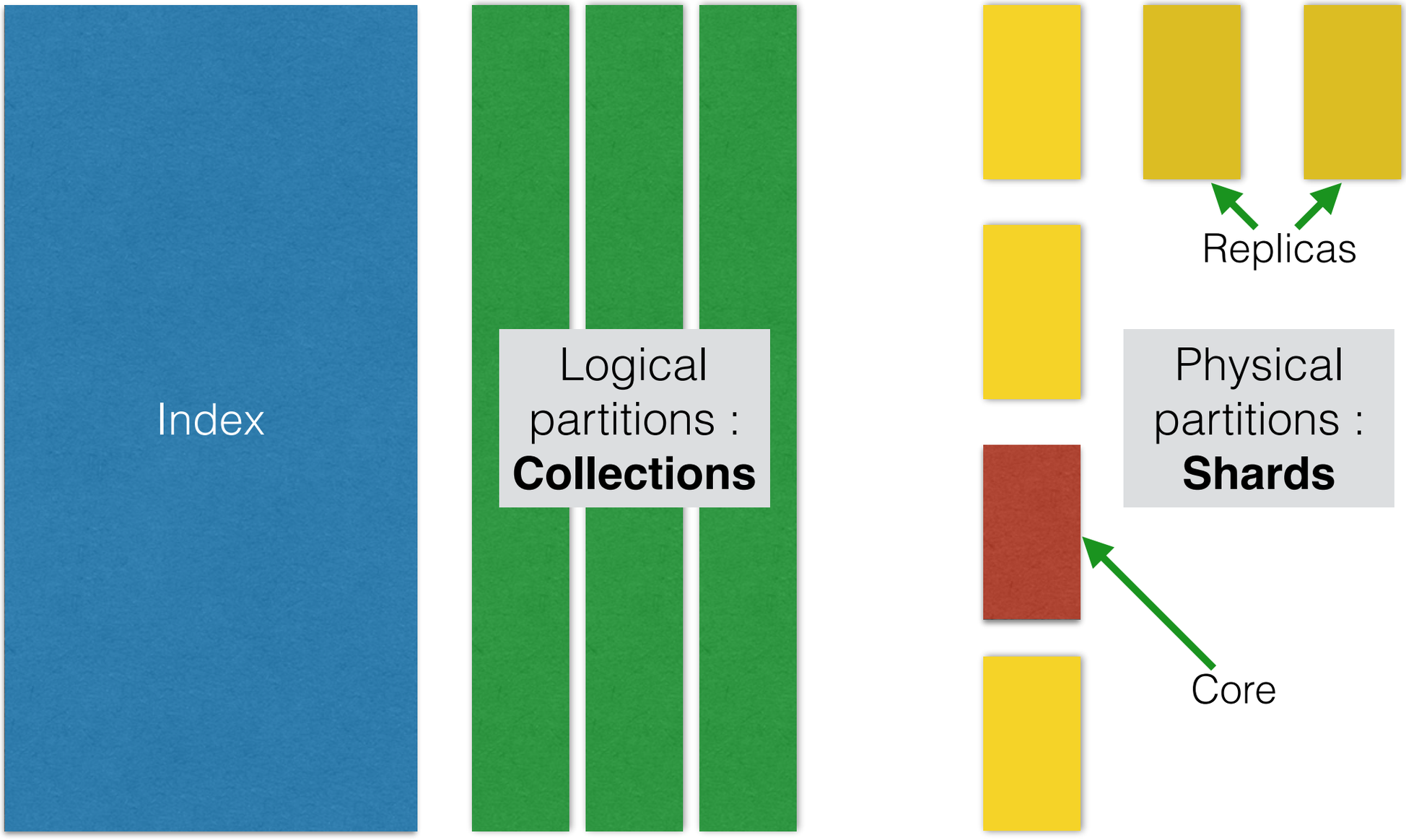
Background : Why Solr?

- We will be using Solr for all projects
  - Production quality search engine with granular programmatic access
  - Robust and reliable - can easily import / merge indices
  - Extensive distributed IR support

Solr Installation & Terminology

- Hopefully, everyone has downloaded and can run Solr
- Currently (if using the solr command) running using pre-packaged Jetty server
    - Will suffice for Project 1. May need revisiting in Project 3 or 4.
- Runs in one of two modes : standalone or cloud
    - Standalone : Single instance, no distributed indexing / querying
    - Cloud : Distributed mode, a given index may be broken down into multiple partitions and/or replicas
    - Zookeeper : Responsible for managing distributed instances
- We now define the following:
    - Directories (self explanatory): Solr home, Data home, Log home
    - Terms : Core, Collection, Shard

Terminology (contd)



Indexing in Solr

Schemaless mode to defining schemas

Quick Start : Let's index some tweets!

- Preparation:
    - Some crawled tweets in JSON format. If none, you can use sample_tweets.json from Resources in piazza.
    - Solr downloaded, but not run in schemaless mode. If you have already done this, see *instructions to clear* below.
    - Your favorite text editor with search/replace functionality
- Solr prep (if you've previously run in schemaless mode)
    - Start solr bin/solr start -e schemaless. This will launch solr on port 8983 with the schemaless example loaded as "gettingstarted" core
    - Navigate to Core Admin on Solr dashboard, "Unload" "gettingstarted" core
    - Stop Solr using bin/solr stop
    - Delete SOLR_HOME/example/schemaless/**
- Minor file edit : search for all instances of 'color":"' and replace with 'color":"#'

Observations

- Solr “guessed” the field types and indexed them
    - Why do you think we needed to edit the color fields?
- Let’s see some of the fields we are interested in for project 1
    - text (tweet_text) : Has been indexed as-is i.e. a full string. No filtering of any sort!
    - lang (tweet_lang) : Language from Twitter, can be used as-is except a minor rename perhaps?
    - entities.urls.url, entities.user_mentions.screen_name, etc.
- For every indexed field, following (notable) attributes can be seen:
    - Field-Type
    - Properties : Indexed, Stored, Multivalued
    - Index analyzer and Query analyzer
    - We discuss what these mean in the following slides.

Solr Schema

- Every *document* is indexed according to a **schema**
- Every document is but a collection of **fields**
- Properties of fields
    - Every field is attached to a native data type (Int, String, etc)
    - A field may be **required** or optional - Solr will reject any documents that don't satisfy this
    - A field may be single valued or **multi-valued** - can you give some examples?
    - An **indexed** field is one that is broken down and stored as tokens, cannot be viewed in its original form
    - A **stored** field is stored as-is and usually used for display purposes
    - A field can be both stored and indexed

Tokenizers, Analyzers and Filters

- A ***tokenizer*** is responsible for breaking down a field into tokens
  - Internally represented as a *token stream*
  - Different operations can thus split, combine, omit or transform tokens from the stream (***filters***)
    - Whitespace tokenizer splits on whitespace, Ngram tokenizer tokenizes on n-grams, etc
- An analyzer is a pre-defined processing chain that combines tokenizers and filters
- Some filter types include Stemmers, Stopword filters, lowercase filters etc.
- Biggest component in the project is configuring Solr correctly!

Additional properties and types

- Solr supports two more types of functionalities related to fields
  - Copy fields : Process the same field differently and/or combine multiple fields into one
    - Use cases : Show names on results as Last Name, First Name but searchable as First name Last name
    - Combine all fields into one default search field
  - Dynamic fields : Specify an analysis for fields with given prefixes or suffixes
    - Useful when schema is not known in entirety upfront
    - Can add new fields on-the-fly without having to specify schema

Modifying the schema

- So far, we've been using the schema less mode.
- You can instead start with a hand-crafted schema.
  - When you add a new core, you need to create a cone directory with your schema.xml and solrconfig.xml
  - You could pick these from one of the examples, modify them and go from there
- Alternatively, you can define new fields in the schema UI
  - Let’s add a new tweet_text field that splits the tweet text

Tips, Tricks & Resources

FAQs, How-Tos etc

Tricky points in Project 1

- We think you might get stuck (not an exhaustive list) at the following points:
    - Removing hashtags, emoticons+, urls, etc from text_xx fields (Pattern Replace filter?)
    - Copying language specific fields (language identification?)
    - Retweet % and volumes (de-duplication?)
    - Emojis, kaomojis, emoticons
    - Date formatting

Help, I'm stuck!

- Any schema change would (most likely) require re-indexing
    - Always verify correct file picked up using the "files" page
- Verify indexing works as expected using "Analysis" page
- Configure logging levels and view logs
- Use the Query, Schema pages judiciously

Resources

- Documents, Fields and Schema Design
- Analyzers, Tokenizers and Filters
- Data Import Handler
- Atomic Updates
- De-Duplication
- Update Request Processors - see section called "Update Request Processor Factories"