EC2 and Solr

For hands-on and lab sessions

Archita Pathak
University at Buffalo (SUNY)

Agenda

- Hands-on
  - Project 1 Overview
  - EC2 instance setup on AWS
  - Solr setup on EC2 instance
- Lab
  - Basic Solr operations and getting familiar with Solr admin console
  - Getting familiar with schema
  - Working with standalone core
    - Creating core
    - Modifying schema

Pre-requisite

- AWS educate account
- For Windows users,
  - Putty
- FileZilla (Optional)

Project 1 Overview

- Tweets from following person of interest
    - Politicians
    - Journalists
    - Social Activists
- From following countries:
    - USA
    - India
    - Brazil
- And following languages
    - English
    - Hindi
    - Portuguese
- Select <u>15 persons of interest</u>, <u>5 from each country</u>
    - Verified
    - Heavy engagement on their tweets

Project 1 Overview

- **Task – 1: Crawling**
  - At least 33,000 tweets in total with not more than 15% being retweets.
  - At least 1000 tweets per person of interest. Note that Twitter allows max 3200 recent tweets to be extracted from a person's account.
  - At least 20 replies to each tweet posted by person of interest in 5 consecutive days.
  - At least 3,000 replies in total across all POIs
  - At least 5,000 tweets per country.
  - At least 5,000 tweets per language i.e, English, Hindi and Portuguese
  - At least 1000 tweets containing hashtags/key words related to the person of interest
- **Note:** Based on how you setup your indexing, you may lose some tweets and/or have duplicates that may reduce your indexed volumes. Hence, it is encouraged that you accommodate some buffer during the crawling stage.

Project 1 Overview

- **Task – 2: Indexing**
  - Person of Interest: Name and Ids of at least 15 persons of interest
  - Country: one amongst USA, India and Brazil
  - All the replies to a particular tweet of a particular person of interest
  - One copy of the tweet text that retains all content (see below) irrespective of the language. This field should be set as the default field while searching.
  - Language of the tweet (as identified by Twitter) and a
  - Language specific copy of the tweet text that removes all stopwords (language specific), punctuation, emoticons, emojis, kaomojis, hashtags, mentions, URLs and other Twitter discourse tokens. Thus, you would have at least four separate fields that index the tweet text, the general field above plus four for each language. For any given tweet, only two of the four fields would have a value.
  - Separate fields that index: hashtags, mentions, URLs, tweet creation date, emoticons+ (emoticons + emojis+ kaomojis)
  - Additionally, geolocation (if present), and any other fields you may like.

Project 1 Overview

- **Solr Field Names**
  1. poi_name : Screen name of one of the 15 persons of interest
  2. poi_id : User Id of one of the 15 persons of interest
  3. verified: Boolean value
  4. country : One of the 3 countries
  5. replied_to_tweet_id : Null for tweet by person of interest else tweet id to which the reply is made.
  6. replied_to_user_id : Null for tweet by person of interest else user id to which the reply is made.
  7. reply_text : Text of the reply to a particular tweet, if replied_to_tweet_id is not null
  8. tweet_text : Default field
  9. tweet_lang : Language of the tweet from Twitter as a two letter code.
  10. text_xx : For language specific fields where xx is at least one amongst en (English), hi (Hindi) and pt (Portuguese)
  11. hashtags : if there are any hashtags within the tweet text
  12. mentions : if there are any mentions within the tweet text
  13. tweet_urls : if there are any urls within the tweet text
  14. tweet_emoticons : if there are any emoticons within the tweet text
  15. tweet_date : Tweet creation date rounded to nearest hour and in GMT
  16. tweet_loc (optional): Geolocation of the tweet. You need to have coordinates in this field.

EC2 instance and Solr setup

- Refer SetupGuidebook.pdf

Solr Basic Operations

- Preparations
  - Use sample.json from Resources in Piazza
  - Your favorite text editor
- Solr Prep (if you've previously run in schemaless mode and posted some data):
  - Start *solr bin/solr start -e schemaless*. This will launch solr on port 8983 with the schemaless example loaded as “gettingstarted” core

Why?

  - 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": "#

Why?

- Transfer sample.json to EC2:
  - *scp -i [path-to-your-identifier.pem] [file-to-be-copied] ubuntu@[public-IP]:/home/ubuntu/solr-8.2.0*

Solr Basic Operations

- Post/index tweets!
  - bin/post –c gettingstarted sample.json
- 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
  - full_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.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, etc.
- Biggest component in the project is configuring Solr correctly!

Customizing Solr

- Coming back to the question from before.
    - Why did we “unload” gettingstarted core which had some previously indexed xml files about techproducts?

- In schemaless mode, solr can **update** your schema based on every new filed it sees. When you post raw tweets on top of techproducts data, it will merge fields of both type of the documents which may lead to querying and analyzing issues.

Customizing Solr

- We have worked on schemaless mode
  - Solr automatically changes schema based on the data
  - You can’t use hand-crafted, modified schema

- Let’s work with our own core now!
  - Adding new core – requires entire conical directory system
  - Hand-crafting schema – will start by using schema generated by Solr in schemaless mode

Creating Core

- **Note:** If you start your solr with -e flag, solr_home sets to *solr 8.2/example/schemaless/solr*. If you don't use -e flag while starting solr, solr_home will set itself to *solr-8.2/server/solr*.
- To define a new core, say IRF19P1, create following directories under solr_home:
  - IRF19P1
  - Navigate to IRF19P1, create "conf" and "data" directories.
- Copy all directories/sub-directories from *solr_home/gettingstarted/conf* to *solr_home/IRF19P1/conf*.
- Rename managed-schema to schema.xml and make your changes.
- Navigate to Core Admin on Solr dashboard. Click on "Add Core". Change both "name" and "instanceDir" to IRF19P1. Click on "Add Core"

Meeting Project Requirements

- Some fields come pre-populated (lang, user.id, in_reply_to_user_id etc.), others will need some additional operation (date, emoticons etc.)
- As per project requirement, you can
  - Copy fields
  - Modify fields/field types
  - Add new fields
- After modifying schema, re-index to see the changes.

Tips and Tricks

- Removing hashtags, emoticons, urls etc. from text_xx fields (Hint: Pattern Replace Filter)
- Copying language specific fields (Hint: language identification)
- Retweet% and volume (Hint: de-duplication)
- Date formatting (Hint: Working with dates and Update Request Processor)
- Any schema change would (most likely) require **re-indexing (re-posting tweets)**
- Always verify correct file picked up by using "Files" tab under your core on Solr dashboard
- Verify indexing works as expected using "Analysis" tab under your core on Solr dashboard
- View logs to debug your Solr
- Use the "Query" and "Schema" tabs judiciously
- Some resources which will help you:
    - Documents, Fields and Schema Design
    - Analyzers, Tokenizers and Filters
    - Data Import Handler
    - Atomic Updates
    - De-Duplication
    - Update Request Processor – see section called "Update Request Processor Factories"