

# COMP9313 - 2019T2

## Assignment #3: Data Curation + Indexing with ElasticSearch

### 25 points total

#### 1 Problem Statement



You are given a corpus consisting of a set of legal case reports represented as XML documents and you are asked to create an index that can support fast search across such case reports. Searches can be done based on the text contained in the document as well as based on specific entity types (e.g., person or organization).

In the first scenario above (searching for text contained in the document) your solution must allow for searching for documents containing general terms such as “criminal law” and “arrest”. In the second scenario, your solution must allow for searching for documents containing specific entities (of a given type) such as organizations (e.g., “Acme Bank”) and person (e.g., “John Smith”).

In order to support queries of the second type (based on entity types), you will need to enrich the original reports with annotations that include mentions (found in the legal case reports) to persons, locations and organizations. The later will require you to use an Entity Recognition API that we will explain later in this document.

#### 2 Requirements



The indexing and search engine to be used for the solution to the problem above must be ElasticSearch (version 6.1.\*). Your program must be developed as a Spark standalone Application (using Scala), to be built using `sbt` as building tool. Your Spark standalone program must receive one parameter in input (in its `args`), which consists in the full path of the directory containing the list of legal case report files (XML files).

You are asked to store all documents in a single index (named `legal_idx`), with a single type (named `cases`). It is up to you to decide of the mapping (schema) to be used for this index. Make sure your solution is able to answer the queries explained above (queries based on entity types as well as queries based on general terms). Below we show two examples:

*Example: Queries based on entity type*

```
$ curl -X GET "http://localhost:9200/legal_idx/cases/_search?pretty&q=location:Melbourne"
$ curl -X GET "http://localhost:9200/legal_idx/cases/_search?pretty&q=person:John"
```

*Example: Query based on general terms*

```
$ curl -X GET "http://localhost:9200/legal_idx/cases/_search?pretty&q=(criminal AND law)"
```

#### 2 What your program must do


Your standalone Spark application must:



- Create the index (in ElasticSearch) with the corresponding mapping (as specified above)
- Perform the necessary data curation tasks to transform the input data into the representation used for indexing (i.e., XML to JSON representation mapping), and enrich the original data with the three entity types specified above (i.e., person, location and organization)
- Index the curated/enriched data using ElasticSearch


### 3 Input

The input for this assignment consists in a list of legal case reports represented as XML files. Each XML file follows the schema below:



```
<?xml version="1.0"?>
<case>
  <name>Sharman Networks Ltd v...</name>
  <AustLII>http://www.austlii.edu.au/au/cases...</AustLII>
  <catchphrases>
    <catchphrase>application for leave to appeal...</catchphrase>
    <catchphrase>authorisation of multiple infringements...</catchphrase>
    ...
  <sentences>
    <sentence id="s0">Background to the current application...</sentence>
    <sentence id="s1">1 The applicants Sharman Networks...</sentence>
    ...
  </sentences>
</case>
```


The schema above shows that a case is made of:

- A name (<name>): This is the title of the case
  - Source URL (<AustLII>): The original source of the legal report
  - A list of catchphrases (<catchphrases>): These are short sentences that summarize the case
  - Sentences (<sentences>): The list of sentences contained in the legal case report
- 

You can find a list of legal report cases in the link below (we use a small excerpt from UCI Machine Learning Repository<sup>1</sup>):

<https://webcms3.cse.unsw.edu.au/COMP9313/19T2/resources/29468>


### 4 Output



The final output consists in an ElasticSearch index of legal report cases enriched with the entity type annotations as discussed previously.

The resulting index must be able to respond to queries (using ElasticSearch's search APIs) involving the entity types discussed previously (i.e., person, location, organization) as well as other queries based on general terms (e.g., search for documents that contains the term "criminal law", search for words appearing in specific properties of the index, etc.).

### 5 Named Entity Recognition API



An important part of enriching the legal reports above is the ability to recognize whether a given term is a mention of one of the entity types of interest (i.e., person, location, organization). For example, in the sentence "*The person went to the Apple store located in...*", the entity recognition task must be able to identify that the word "Apple" in this sentence refers to an organization (not a fruit).

For this assignment, you must use the Stanford Core NLP server to help you recognize named entities. You can download the server and get the full documentation of the tool from the link below:

<https://stanfordnlp.github.io/CoreNLP/corenlp-server.html>

The Stanford Core NLP server allows you to get access to a number of NLP task through web APIs (HTTP requests). One such task is that of [Named Entity Recognition](#) (or NER):

---

<sup>1</sup> <https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports>



Notice that Stanford Core NLP can be used both as a library inside your Java code or as a server. In this assignment, we will use the Core NLP Server and access the NER functionalities through HTTP requests. This means that you will have to download the sever and run it locally on your computer to get access to the Named Entity Recognition functionality through web APIs (HTTP requests).

Since you will rely on Core NLP for recognizing named-entities, the ability of recognizing person, location and organization will be limited to what is provided by Core NLP (the performance of the entity recognizer will not be part of the assessment of this assignment).

## 6 ElasticSearch and Core NLP hostnames / ports



Make sure your ElasticSearch server is running on `http://localhost:9200` and that your Core NLP server is running on `http://localhost:9000` (these are their default ports).

Use of these addresses/ports are mandatory, as for assessment, we will assume these servers are running in these hostnames / ports.

## 7 Running your Program



In order to assess your solution, we will build your program using `sbt`. We will run your program using `spark-submit`, where we will provide the directory containing the legal case report files (XML files) as argument:

```
$ spark-submit --class "CaseIndex" --master local[2] \
  JAR_FILE FULL_PATH_OF_DIRECTORY_WITH_CASE_FILES
```

Notice that you must use the name `CaseIndex` for the object containing your main program.

Make sure your solution can be built and run on VLab (notice that VLab uses Spark 2.4.3 and Scala 2.11.12)

## 8 Assignment Submission

**Deadline:** 04 August 2019 20:59:59

Log in to any CSE server (e.g. williams or wagner) and use the *give command* below to submit your solution:

```
$ give cs9313 assignment3 z9999999.zip
```



where you must replace `z9999999` above with your own `zID`. The zip file above must contain the following:

- Your project directory (which we will use to build your program). Do not include the subdirectories `project` and `target`, as these typically occupy lots of space. Include only the `build.sbt` file and your `src` subdirectory.
- A PDF document, named `assignment3_solution.pdf` (maximum 2 page, 10 points font-size Arial), that explains your index design and solution implementation (use of figures is highly encouraged to explain your solution). This PDF must also include example queries that show your index in action for general term searches and entity based searches (for all three entity types). Provide queries using the `curl` command as used in Lab #5 (use only search based on query strings (not DSL)).

You can also submit your solution using WebCMS, or Give:

<https://cgi.cse.unsw.edu.au/~give/Student/give.php>

If you submit your assignment more than once, the last submission will replace the previous one. The late submission penalty (below) will be applied based on the timestamp of your last submission. To prove successful submission, please take a screenshot and keep it for your own record. If you face

any problem while submitting your code, please e-mail the Course Admin (Maisie Badami, [m.badami@student.unsw.edu.au](mailto:m.badami@student.unsw.edu.au))

## 9 Late submission penalty

10% reduction of your marks for the 1st day, 30% reduction/day for the following days.

## 10 Assessment

Your source code will be manually inspected and marked based on readability and ease of understanding. We will run your code to verify that it produces correct results. The code documentation (i.e. comments in your source code) and solution explanation (PDF document) are also important. Below, we provide an indicative assessment scheme (maximum mark: 25 points):

Solution implementation and resulting index	15 points
Documentation (PDF document)	5 points
Code structure and source code documentation (comments)	5 points

## 11 Plagiarism

This is an *individual assignment*. The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined manually.

Do not provide or show your assignment work to any other person - apart from the teaching staff of this course. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted, you may be penalized, even if the work was submitted without your knowledge or consent.

*Reminder:* Plagiarism is [defined as](#) using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- [Plagiarism and Academic Integrity](#)
- [UNSW Plagiarism Procedure](#)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible for ensuring that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository, if using one (e.g., Github and similar). Note also that plagiarism includes paying or asking another person to do a piece of work for you and then submitting it as your own work.

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW.