BIG DATA TECHNOLOGIES: ITEC874

MACQUARIE UNIVERISTY

Assignment 2: Processing Big Data and Knowledge Lake

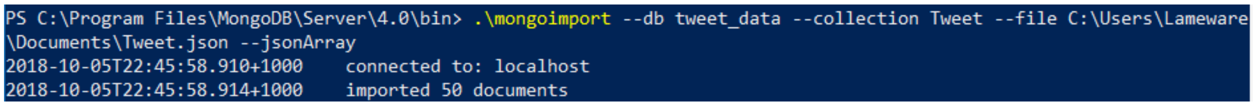
Submitted by: Sukhdeep Singh (44442467)

Q1.

Importing the data steps:

Command used: C: \Program Files\ MongoDB\ Server\ 4.0\ bin > .\mongoimport--db tweet_data--collection Tweet--file C: \Users\ Lameware\ Documents\ Tweet.json--jsonArray

Output produced:



```
PS C:\Program Files\MongoDB\Server\4.0\bin> .\mongoimport --db tweet_data --collection Tweet --file C:\Users\Lameware
\Documents\Tweet.json --jsonArray
2018-10-05T22:45:58.910+1000    connected to: localhost
2018-10-05T22:45:58.914+1000    imported 50 documents
```

2. Connecting to mongo: .\mongo.exe

3. Select DB: use tweet_data

A) Write a MongoDB query that returns all the Tweets : db.Tweet.find().pretty()

B) Write a MongoDB query to find one of your Tweets by name: db.Tweet.find({"user.name":"user 03"}).pretty()

C) Update your two favourite Tweets to have two tags called 'My number 1 Tweet' and 'My number 2 Tweet'. Show two ways to do this. Do the first using update() and do the second using save().

1. db.Tweet.update({ "user.name":"user 03"},{ $set:{"entities.hashtags":"My Tweet number 1"}})

2. var mongo = db.Tweet.findOne({"user.name":"user 24"});
   mongo["entities"]["hashtags"]="My tweet number 2"

   db.Tweet.save(mongo)

D) Write a MongoDB query that returns only Tweets that have tags: db.Tweet.find({"entities.hashtags":{$ne:[]}}).pretty()

E) Write a MongoDB query to find the Tweet Id for those Tweets which contain the keyword 'health' in their text: db.Tweet.find({"text":{$regex:/health/i}},{id_str:1}).pretty()

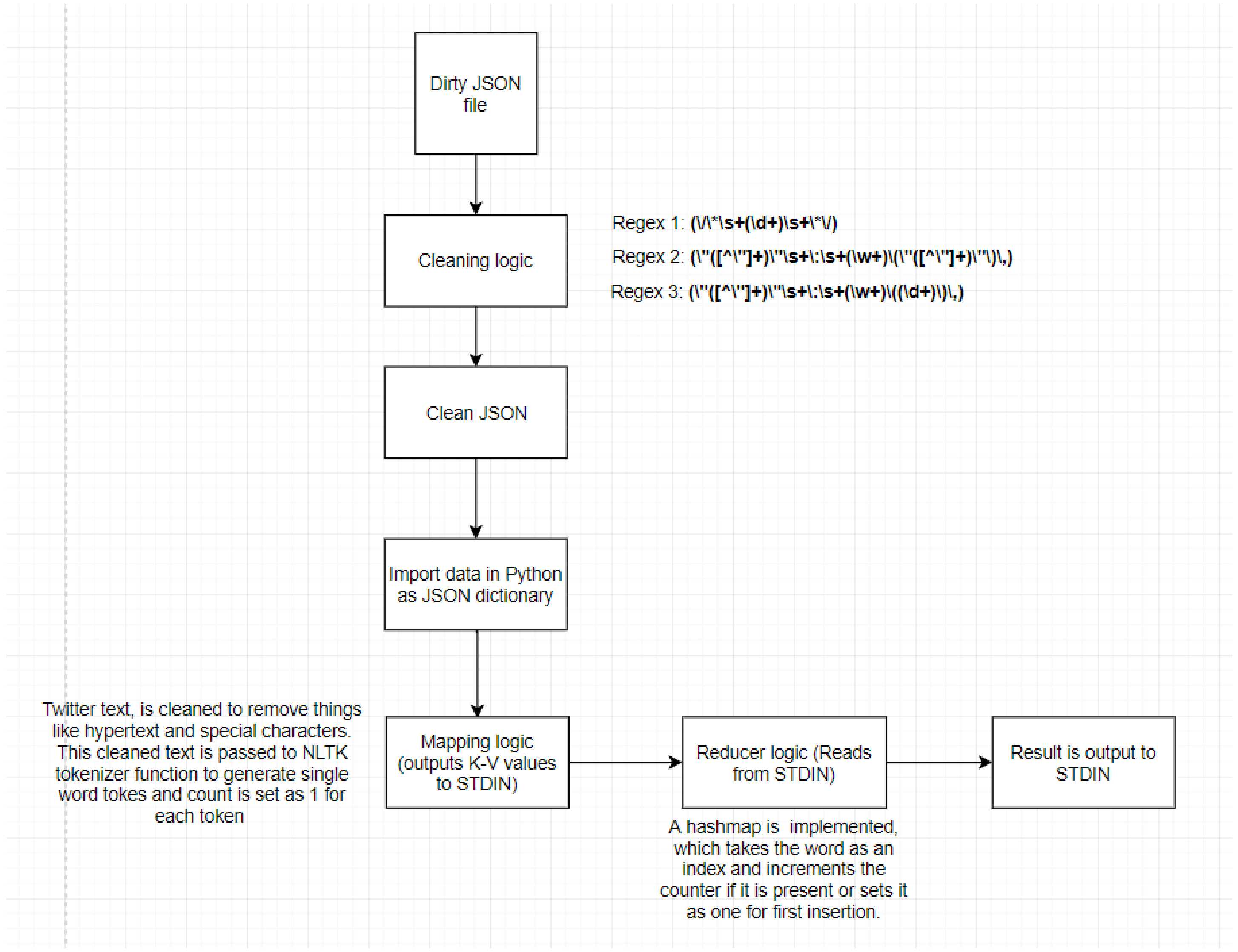
Q2)

Data cleaning: Since the provided JSON was invalid, it had to be pre-processed first before we can use map-reduce

1. Square brackets added to beginning and end of file.
2. Text of the pattern /* digits */ replaced with comma using the regex **(\/*\s+(\d+)\s+\*\/) [Find and replace tool in notepad ++]** first occurrence of the pattern i.e /* 1 */ was deleted manually.
3. Key : _id was removed from the file using regex: **(\"([^\"]+)\"\s+\:\s+(\w+)\(\"([^\"]+)\"\)\,) [Find and replace tool in notepad++]**

4. Key : media.id was removed from the file using regex: **(\"([^\"]+)\"\s+\:\s+(\w+)\((\d+)\)\,) [Find and replace tool in notepad++]**
5. Json was validated by importing it into mongo collection.

A) MapReduce: Calculate the count of number of occurrences of each word in the text of Tweets


Dirty JSON file
Cleaning logic
Regex 1: (\/*\s+(\d+)\s+\*\/)
Regex 2: (\"([^\"]+)\"\s+\:\s+(\w+)\(\"([^\"]+)\"\)\,)
Regex 3: (\"([^\"]+)\"\s+\:\s+(\w+)\((\d+)\)\,)
Clean JSON
Import data in Python as JSON dictionary
Twitter text, is cleaned to remove things like hypertext and special characters. This cleaned text is passed to NLTK tokenizer function to generate single word tokes and count is set as 1 for each token
Mapping logic (outputs K-V values to STDIN)
Reducer logic (Reads from STDIN)
A hashmap is implemented, which takes the word as an index and increments the counter if it is present or sets it as one for first insertion.
Result is output to STDIN

```
Mapping logic:


import json
import sys
import nltk
import re


#Loading the clean Json file

with open('10000 tweets_old.json', 'r', encoding="utf8") as f:
    data = json.load(f)


#Iterating over the keys of the json dicionary

for i in data:

    text =i['text'] #This grabs the text key from the tweet

#Here we are passing it to tweet cleaning regex to remove things like special characters
and URL

    text= re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", text)

#Using the NLTK tokenize function to generate tokens of the sentence


    tokens = nltk.word_tokenize(text)

#Outputting the Key Value pairs to STDIN

for word in tokens:

        print("{0}\t{1}".format(word,1))
```

Reducer Logic:

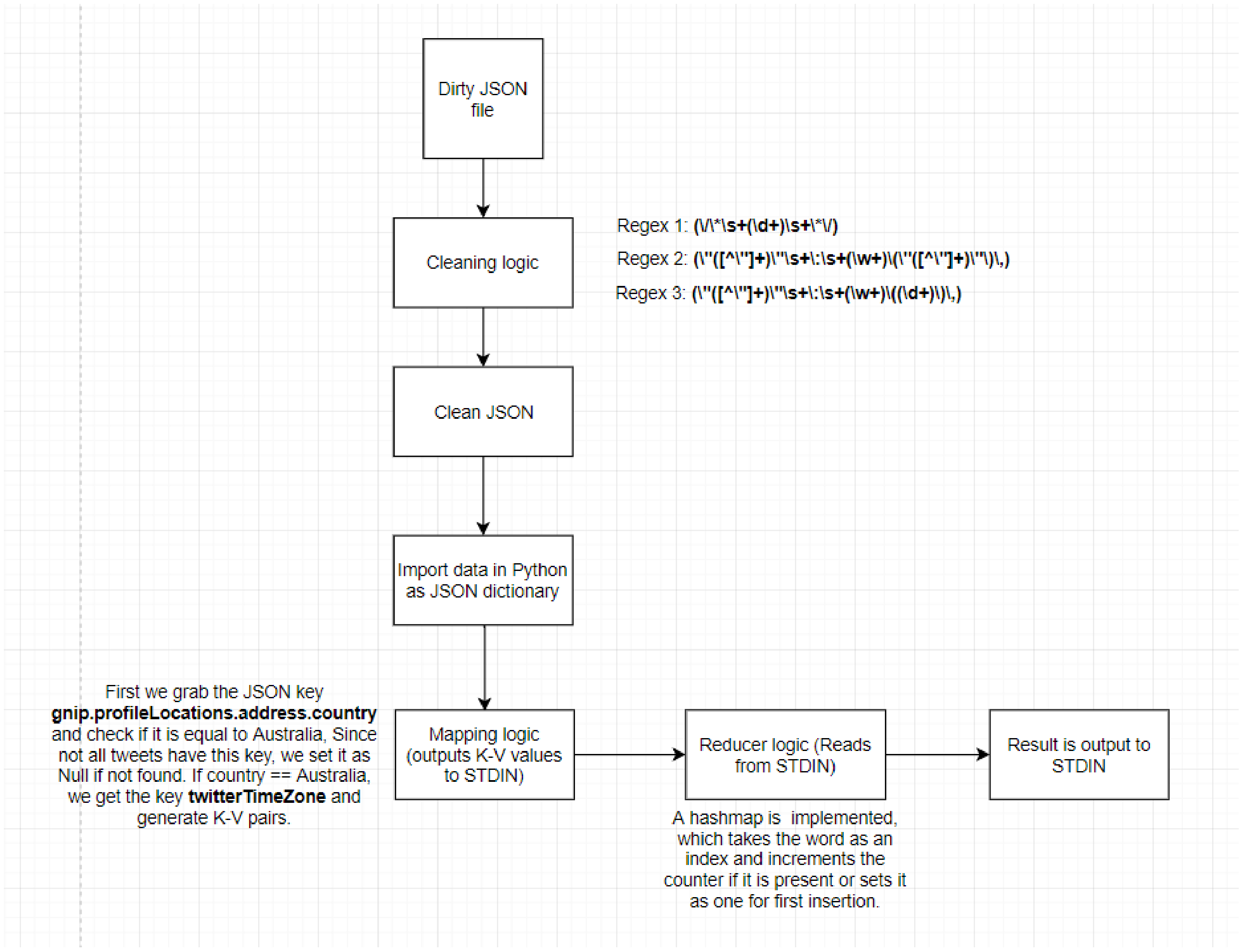
```
import sys

# Creating a hashmap
word2count = {}

# Reading input from stdin


for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        continue

    try:
        word2count[word] = word2count[word]+count
    except:
        word2count[word] = count

# write the tuples to stdout
```

```
for word in word2count.keys():
    print ( word, word2count[word] )
```

Command used: type "10000 tweets_old.json"|python mapper.py|python reducer.py|sort|more

B) Calculate the count of number of tweets for a list of different cities.

Mapper logic:


Dirty JSON file
Cleaning logic
Regex 1: (\/*\s+(\d+)\s+\*\/)
Regex 2: (\"([^\"]+)\"\s+\:\s+(\w+)\(\"([^\"]+)\"\)\,)
Regex 3: (\"([^\"]+)\"\s+\:\s+(\w+)\((\d+)\)\,)
Clean JSON
Import data in Python as JSON dictionary
First we grab the JSON key gnip.profileLocations.address.country and check if it is equal to Australia, Since not all tweets have this key, we set it as Null if not found. If country == Australia, we get the key twitterTimeZone and generate K-V pairs.
Mapping logic (outputs K-V values to STDIN)
Reducer logic (Reads from STDIN)
A hashmap is implemented, which takes the word as an index and increments the counter if it is present or sets it as one for first insertion.
Result is output to STDIN

```
import json
import sys
import nltk
import re

#Reading Json file
```

```
with open('10000 tweets_old.json', 'r', encoding="utf8") as f:
    data = json.load(f)




#Iterating over the keys

for i in data:
    try:

        #Here we are catching the country key located under “gnip”

        text= i['gnip']['profileLocations'][0]['address']['country']

        #Since not all tweets have this property we just set it to null if not found

    except KeyError:
        text="Null"
        continue


    #Checking if country obtained is Australia
    if(text=="Australia"):


        #Getting the city
        text= i['actor']['twitterTimeZone']

        #Printing the output to STDIN
        print("{0}\t{1}".format(text,1))
```

Reducer Logic:

```
import sys

# Creating a hashmap
word2count = {}

# Reading input from stdin


for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        continue

    try:
        word2count[word] = word2count[word]+count
    except:
        word2count[word] = count

# write the tuples to stdout
for word in word2count.keys():
    print ( word, word2count[word] )
```

Command used: type "10000 tweets_old.json"|python mappercity.py|python reducer.py|sort|more

Q3)

Data Ingestion: Data ingestion refers to the process of loading, transforming and processing the information. Usually data ingestion process involves multiple heterogeneous data streams. Data ingestion can be real time, synchronous or batched depending on the required use case. Let’s look at some of the existing big data technologies

1. Apache Kafka: Apache kafka is a high throughput messaging system with a highly fault tolerant, low latency platform which is designed for dealing with high bandwidth data pipes. Kafka is distributed, partitioned and replicated commit log service which serves as a central data bank for various organizational needs.

2. Apache Spark: Apache spark is described as a unified analytics engine for big data and ML. Spark uses in-memory storage to speed up the processing time works of tasks, Spark also circumvents the linear dataflow by allowing for a flexible pipeline for industry specific use case. Most of the Data Science algorithm are iterative in nature, therefore Spark’s ability to cache the datasets in memory provides a boost to calculation speeds. Sample use case: Analyze intraday stock prices, detection of fraudulent transactions in real-time

3. Apache flink: Apache flink is described as an open source stream processing framework, this suitable for cases where data naturally comes in form of never ending stream, such as length of a session, IOT data, and customer activities. Unlike Hadoop and Mapreduce, stream processing allows us to respond to data very quickly (milliseconds)

4. Wavefront: Wavefront is a cloud based solution for ingestion, visualization and generation of alerts based on metrics. Wavefront allows engineers to ingest millions of datapoints every second and allows manipulation of timeseries data.

5. Amazon Kinesis: Kinesis is a fully managed cloud based service which provides platform for analyzing terabyte scale real time data streams from hundreds of different sources.

Data Cleaning: Data cleansing or data cleaning is the path toward perceiving and modifying ruffian or misguided records from a record set, table, or database and recognizing divided, wrong or immaterial parts of the data and changing, or deleting the dirty or coarse data. Data cleansing may be performed naturally with data wrangling instruments, or as bunch taking care of through scripting.

1. Drake: is an easy to use content based workflow based tool which helps in the organization of the command execution around the data and its dependencies, Drake allows the users to define the data processing steps along with the respective inputs and determines which commands to execute.

2. Open Refine: OpenRefine is a extensive tool for working with dirty data: cleaning it; modifying it from one format into another; and extending it with web services and external data.

3. Data Wrangler: Data wrangler is cloud based service which enables data engineers to clean and plan chaotic data quickly and precisely. When you import datasets to Wrangler, it starts to arrange and structure your information as it arrives. Wrangler will then propose normal changes and conglomerations.

4. DataCleaner: The core of DataCleaner is a solid information profiling engine for finding and breaking down the nature of dataset. It helps to discover the trends and various attributes of your information streams.

5. Winpure Data cleaning Tool: This solution helps to clean, correct, standardize and remove all duplications across the datasets. The Data clean atrix provides a simple yet sophisticated method for applying a multitude of cleaning techniques to the dataset.

Data Integration: Data integration refers to the process of combining the data residing in various sources and providing user with unified view of the same. Some of the tools used for data integration.

1. Actian: This is a database integration software solution. Whether you are designing an integration processes for data warehouses for loading, converting data between various formats or deploying complex application integration scenarios. Actian Data Integrator includes numerous foundation features that accelerate the creation of flexible integration solutions.

2. Adptia: Adeptia Integration Suite (AIS) is an enterprise-class solution that simplifies all aspects of cloud and on-premise integration including B2B Integration, Application Integration (ESB), Business Process Management, and Data Integration (ETL).

3. Built.io: Built.io offers a cloud-based Integration PaaS product called Flow. It is available in two editions (Express for business users and Enterprise for technical teams) with the ability to export Flows and share across each of them as an end-to-end integration solution.

4. Dell boomi: This is an integration solution built in the cloud Organizations of all sizes—from small businesses to the largest global enterprises—trust Dell Boomi to connect any combination of cloud and on-premise applications.

5. HVR: HVR provides real-time data replication for Business Intelligence, Big Data, and hybrid cloud. HVR can solve for a variety of Data Integration use cases including data migrations, Data Lake consolidation, geographic replication, database replication, and cloud integrations.

Data and Knowledge Extraction: Knowledge extraction is the process of generation of facts/knowledge from organized (social databases, XML) and unstructured (content, archives, pictures) sources. The subsequent knowledge should be in a machine-discernable and machine-interpretable arrangement and must speak to knowledge in a way that encourages inferencing.

1. DBpedia Spotlight: DBpedia Spotlight is a tool for annotating mentions of DBpedia resources in text. This allows linking unstructured information sources to the Linked Open Data cloud through DBpedia. DBpedia Spotlight performs named entity extraction, including entity detection and name resolution (in other words, disambiguation).

2. OpenCalais: Calais is a service by Thomson Reuters that automatically extracts semantic information from web pages in a format that can be used on the semantic web.Calais was launched in January 2008, and is free to use. The Calais Web service reads unstructured text and returns Resource Description Framework

formatted results identifying entities, facts and events within the text.

3. Dandelion dataTXT: Dandelion API allows the user to extract entities from text/URL with a user configurable level of precision, lower the confidence level allows the API to return more number tags and vice versa.

4. Poolparty Extractor: This solution allows the user to perform knowledge extraction, auto classification and entity linking which is automated based on a machine processable framework. This solution allows native integration with any RDF graph based databases.

5. MaKi: MAKI project, the web knowledge extraction aims at using an expert-centric methodology. Following this idea, the whole knowledge extraction task is be designed around the expert and his/her knowledge. From data acquisition to knowledge extraction, the expert is assisted by a set of tools that help her through the process with minimal intervention from the developers.

Data and Knowledge Enrichment: Data enrichment refers that refers to the processes undertaken to the enhance, refine or otherwise add context to the raw data, the ideas and similar concepts which contribute towards making a data set valuable are categorized under the data enrichment.

1. Leadspace: Leadspace looks at electronic records from across the web and builds that data into your existing customer and prospect database. The web data combines with predictive analytics to help you understand which customers are most likely to purchase based on previous customer actions and your picture of the ideal customer.

2. Lead genius: This platform-agnostic data tool can help all of your current lead data platforms run better. Data enrichment finds data for leads based on named accounts or target companies and fills in data that you already have to make it cleaner and more useful. Fill out sparse prospect data for an entire target account, or find new prospects within your current accounts through discovery and enrichment tools.

3. Openrise: Openprise offers data enrichment that fills blank data fields based on current customer information and readily accessible online data. This is achieved by comparing customer data with openly accessible data and matching it with records. Openprise can help you find the right person in the account.

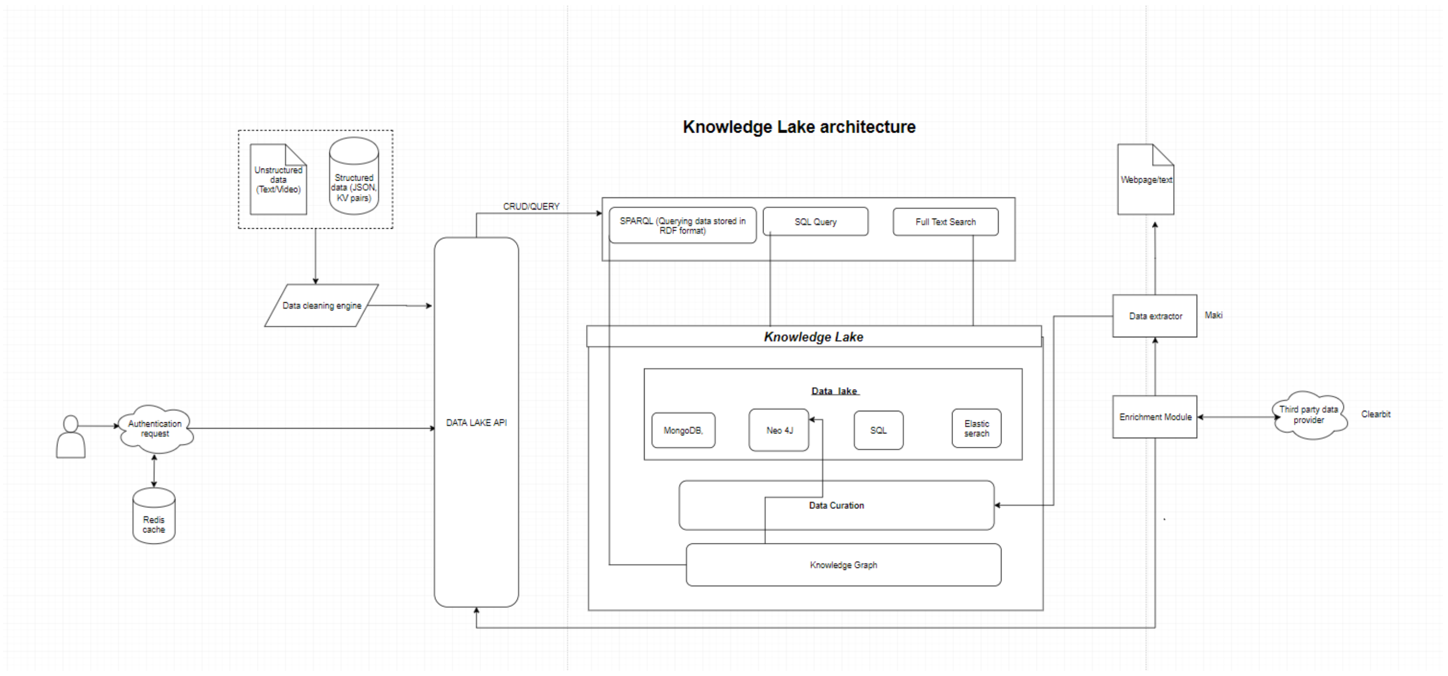
4. InsideView: InsideView merges all the disparate data records to create a single source of truth for generating market leads. Customer data validation, lead enrichment, and data cleansing are some of the attributes.

5. Clearbit: Helps sales team enrich the records of their prospects, and personalization of the marketing campaigns.

Data and Knowledge Linking

1. DBMedia: Given above
2. Wikidata: Based of wiki media foundation which allows linking based on topics and subjects
3. Babelnet:
4. Semantic media wiki:
5. Freebase:

Architecture for Knowledge Lake:



The above figure shows a possible architecture for a Knowledge lake. A Knowledge lake is basically a Data Lake with Contextualized Data. The process of contextualizing raw data is known as Data Curation.

Some of the key data curation services supported by our architecture are :

1. Extraction: This adds value to our data by extracting features such as keywords and parts of speech. Data extraction can be implemented using tools as Maaki(discussed in the tools in Part 3) as shown above.

2. Linking: We link our extracted and enriched data to external knowledge bases like Wikidata and to other contextualized databases.

3. Enrichment: Enrichment is another key step in the process of curating and contextualizing data. We can enrich our extracted features by adding synonyms. The tool that can be used for data enrichment is Clearbit.

4. We use the standard SQL to query structured data from our data lake.

5. We use SPARQL to query contextualized data from the knowledge lake. SARQL is basically a query language, used for RDF data formats.

6. Datalake API is protected behind a security layer, which uses REDIS cache for authentication.

7. Data cleaning engine, houses multiple algorithms and tools for cleaning the data prior to ingestion.