Data Collection

Url : https://kermitt2-grobid.hf.space/api/processReferences

Method:

List of all references was created using `soup.findAll('biblStruct')`
We extracted the venue/journal using `ref.find('title', attrs = {'level':"j"}).get_text()`
Authors were extracted using `ref.findAll('author').` `' '.join(auth.stripped_strings)` was used to concatenate first, middle and last names of authors using a single space.

Output dictionary format: {paper_id_1:{journal_1:[author_1, author_2]}}

The **A3_all_venues.json** file contains the final dictionary.

Setting up Apache Kafka

- `sudo wget https://dlcdn.apache.org/kafka/3.4.0/kafka_2.13-3.4.0.tgz`
- `sudo tar -xvzf kafka_2.13-3.4.0.tgz`
- `cd kafka_2.13-3.4.0/config`
- `sudo nano server.properties`
- `cd ..`
- `sudo bin/zookeeper-server-start.sh config/zookeeper.properties`
- `sudo JMX_PORT=8004 bin/kafka-server-start.sh config/server.properties`
- `sudo bin/cmak -Dconfig.file=conf/application.conf -Dhttp.port=8080`

```
# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=localhost:2181
```

```
# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
advertised.listeners=PLAINTEXT://localhost:9092
```

Creating topics for each venue:

There are a total of 22151 venues in our dataset. For the purpose of demonstration we have written a bash script create_topics.sh to create 100 most commonly occurring topics out of these 22151. The script can be easily modified to generate topics for each venue.
The cluster cluster1 was created using the CMAK GUI.

DGIM algorithm

First of all DGIM_producer generates a stream of 0's and 1's for each of the venues selected. The command line argument n helps you specify how many most commonly occurring venues you want to generate the venue streams for.
The DGIM_producer takes in a command line argument that is the venue name, and processes the stream generated for that topic as per the DGIM algorithm to estimate the number of 1s on the last 500 bits received.

To automate this and process streams for each of the venues, we have a consumer_script_generator utility that takes in n as the commandline argument and creates processes for each of the n most common venues of the dataset.

FM algorithm

The FM_consumer streams each paper's authors to its venue topic. This stream is then processed by the FM_consumer where each element is hashed to a 4 byte value using the hashlib library of python. This hash is then used for updating the global count of max number of trailing zeros in the hash so far and the estimate is printed as per the DGIM algorithm.

Relevant code:

1. Data collection notebook: https://colab.research.google.com/drive/1_NrKpOHpx30xjHgUSrrp0x6tzRG23Siz
2. All the kafka processing files: https://github.com/tanyagupta1/BDA_A4