Visual Analytics – Assignment 2 – Part 2

Claudio Maggioni

Indexing

Similarly to part 1 of the assignment, the first step of indexing is to convert the newly given CSV dataset (stored in `data/restaurants_extended.csv`) into a JSON-lines file which can be directly used as the HTTP request body of Elasticsearch document insertion requests.

The conversion is performed by the script `./convert.sh`. The converted file is stored in the JSON-lines file `data/restaurants_extended.jsonl`.

The sources of `./convert.sh` are listed below:

```
#!/bin/sh
set -e

SCRIPT_DIR=$(cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null && pwd)

input="$SCRIPT_DIR/data/restaurants_extended.csv"
output="$SCRIPT_DIR/data/restaurants_extended.jsonl"


# In order:
# - Convert CSV to JSON
# - Convert JSON array in JSON lines notation
# - Remove last line (which is all `null`)
cat "$input" | jq -s --raw-input --raw-output \
  'split("\n") | .[1:-1] | map(split(",")) |
    map({
      "id": .[0],
      "name": .[1],
      "cityRaw": .[2],
      "city": .[2] | split("/") | .[0],
      "country": .[2] | split("/") | .[1],
      "continent": .[2] | split("/") | .[2],
      "location": {
          "lon": .[8] | sub("^\"\\["; "") | sub("\\s*"; "") | tonumber,
          "lat": .[9] | sub("\\]\"$"; "") | sub("\\s*"; "") | tonumber,
      },
      "averageCostForTwo": .[3],
      "aggregateRating": .[4],
      "ratingText": .[5],
      "votes": .[6],
      "date": .[7]
    })' "$input" | \
        jq -c '.[]' > "$output"
```

The only change in the script is the way the field containing the restaurant location is parsed. In the extended

dataset, city, country and continent are in this field and separated by `/`. The script maps the three values in separate fields and additionally maps the entire string to an additional `cityRaw` field which is used in the generation of the runtime field for part 2.

The sourced of the updated upload script, loading the new index are listed below:

```
#!/bin/bash

set -e

SCRIPT_DIR=$( cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )

elastic_dir="$HOME/bin/elasticsearch-8.6.2"
elastic_url="https://localhost:9200"
crt="$elastic_dir/config/certs/http_ca.crt"

input="$SCRIPT_DIR/data/restaurants_extended.jsonl"
password="GZH*wqNTvQ0WRdrPrpHm"

index_name="restaurants_extended"

# Create index
curl --cacert "$crt" -u "elastic:$password" \
    -X DELETE "$elastic_url/$index_name" | jq . || true
curl --cacert "$crt" -u "elastic:$password" \
    -X PUT "$elastic_url/$index_name" | jq .

# Upload mappings
cat mappings.json | curl --cacert "$crt" -u "elastic:$password" -X POST \
    --data-binary @- "$elastic_url/$index_name/_mappings/" \
    -H "Content-Type: application/json" | jq .

# Upload documents one by one
while IFS= read -r line
do
    id=$(echo "$line" | jq '.id | tonumber')
    echo $line | curl -k --cacert "$crt" -u "elastic:$password" -X PUT \
        --data-binary @- "$elastic_url/$index_name/_doc/$id" \
        -H "Content-Type: application/json" | jq ._id &
done < "$input"
```

Mappings are stored in `mappings.json` and are identical to the ones in Part 1 other than for the new location fields and their `.keyword` counterparts similarly generated as the old `city` field.

9499 documents are imported.

Data Visualization

The Dashboard, Canvas, and requested dependencies (like scripted fields and stored searched) are stored in the JSON object export file `export.ndjson`. Screenshot of the Dashboard and Canvas can be found below.

The scripted field `continent_scripted` has been generated with the following Painless expression:

```
doc['cityRaw.keyword'].value.substring(doc['cityRaw.keyword'].value.lastIndexOf("/") + 1)
```

The expression extracts the last portion of the `cityRaw` field, i.e. the portion of text between the last `/` and the end of the field, which contains the continent.

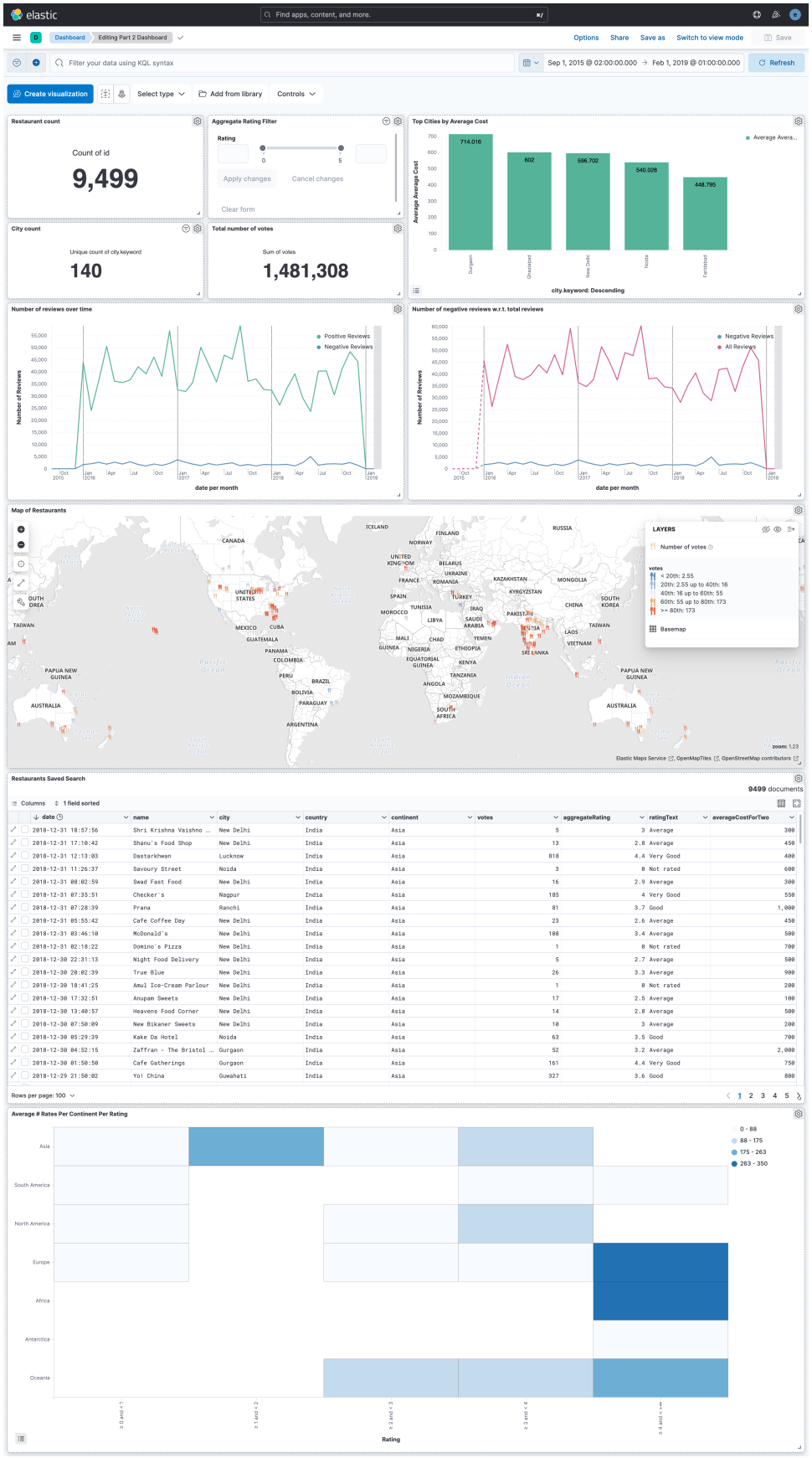

Figure 1: Part 2 Dashboard

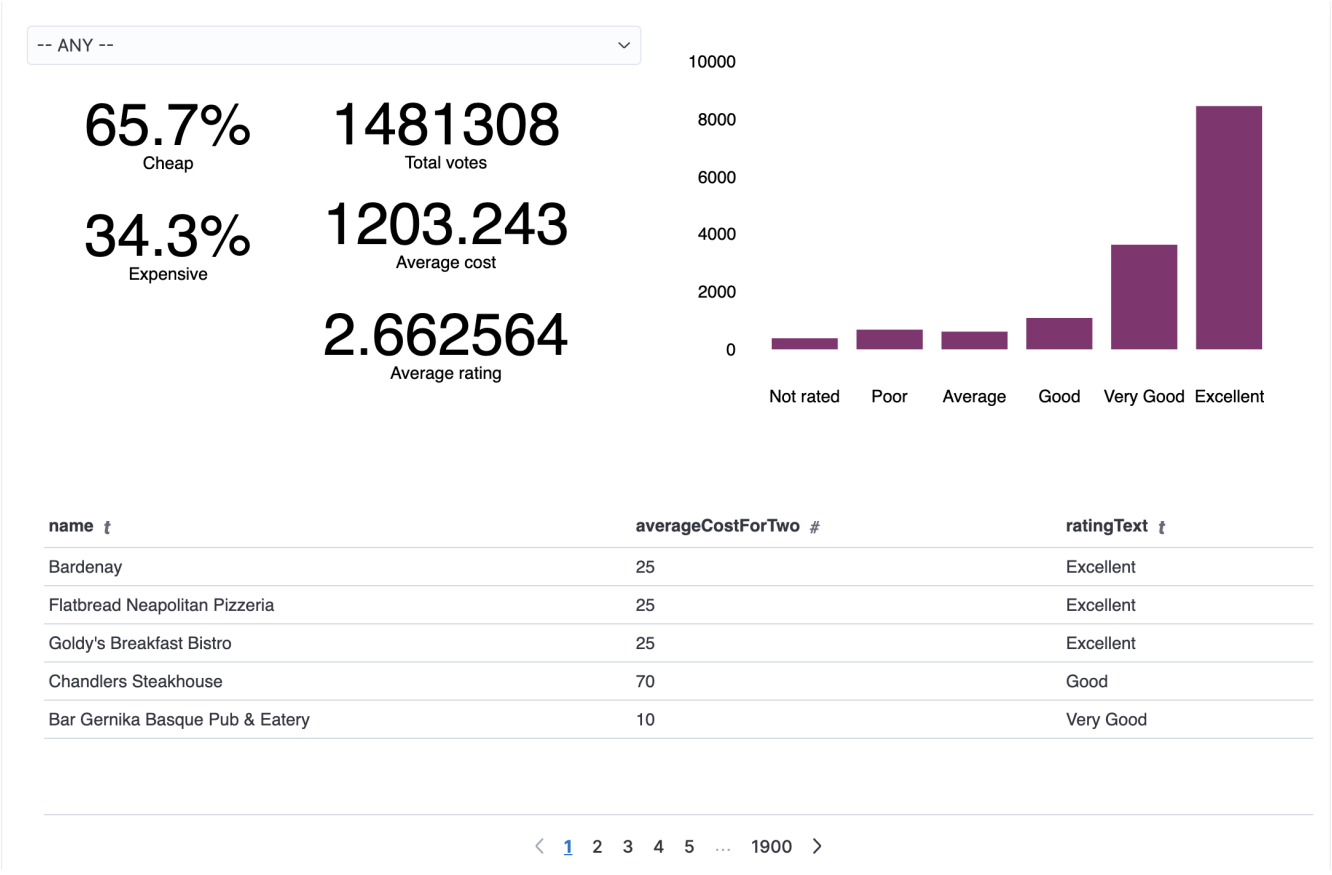

Figure 2: Part 2 Canvas with no city selected

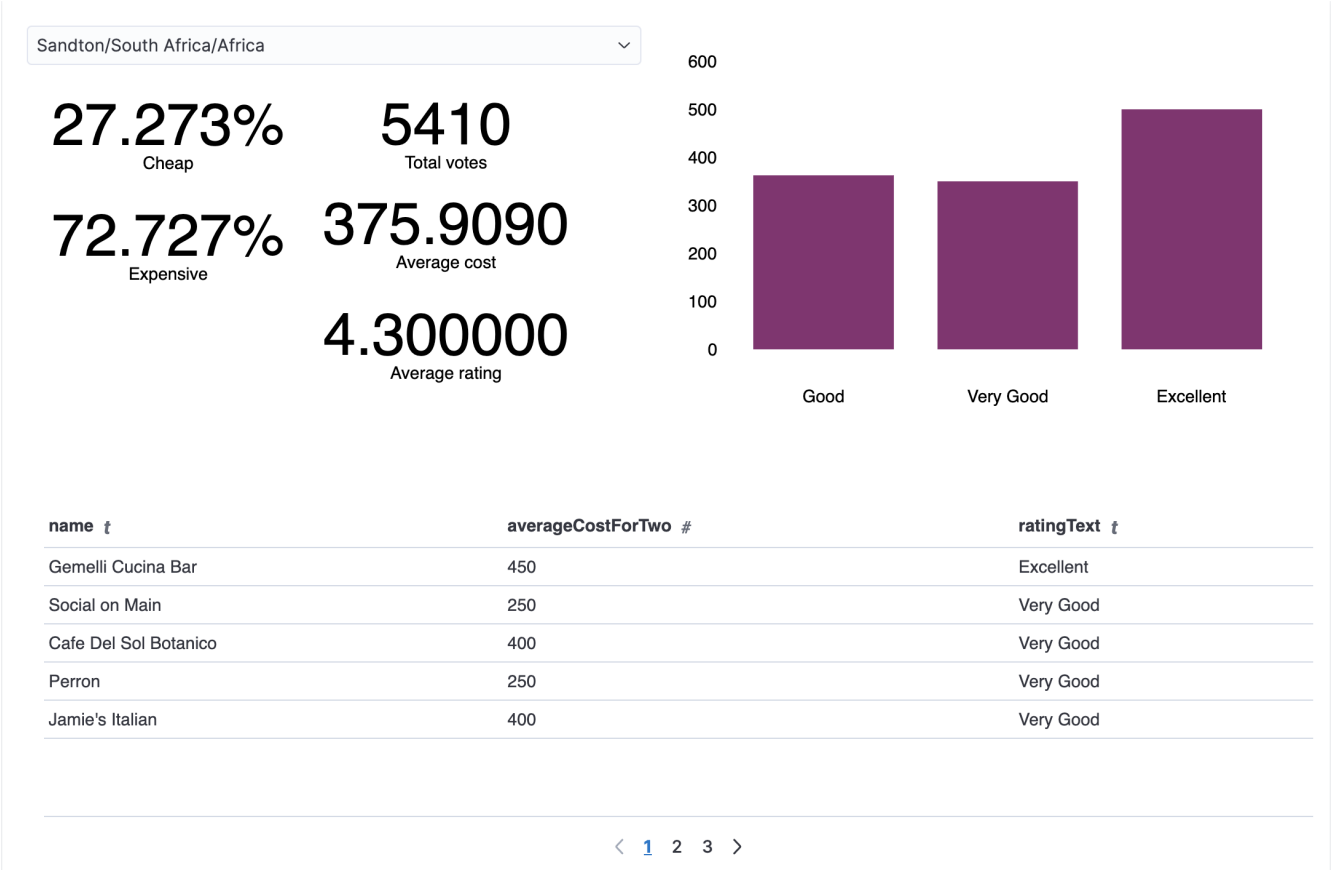

Figure 3: Part 2 Canvas with a city selected

Ingestion Plugin

Sources for the ingestion plugin can be found in the Gitlab repository:

usi-si-teaching/msde/2022-2023/visual-analytics-atelier/elasticsearch-plugin/ingest-lookup-maggicl.

The plugin can be built and installed on Elasticsearch with the script `./install-on-ec.sh` included in the repository by changing the variable `ES_LOCATION` to the path to the local installation of Elasticsearch.

The plugin works as illustrated in the `README.md` file in the repository, and it has been tested with a unit test suite included in its sources.

The plugin lookup procedure works by splitting the indicated field in words (non-empty sequences of non-space characters – according to the PCRE regular expression specification) and matching each word with the given substitution map, performing substitutions when needed.