Lab Assignment 6: Working with the Mongo Shell

In this assignment you will practice more working with MongoDb and its shell. We will be using the text from the Game of Thrones (Song of Ice and Fire) series of books, that can be found here[1].

I have prepared a Python script that you will need to run first. This script downloads the text from those webpages, and organizes it in a big JSON file. You will then write that JSON file to your database. You will need to do the following:

```
curl https://raw.githubusercontent.com/skiadas/DataWranglingCourse/gh-pages/labs/got.py > got
python3 got.py
mongoimport -h ds011168.mlab.com:11168 -d wranging -c got -u haris -p haris --file got.json -
```

You will need to change the name of the server, database and username and password on the third line, to match your database. The first line downloads the python script, the second executes it and creates a got.json file, and the thirs pushes that to the database. The second and third step will take some time.

Now your database has a new collection called got. Log into your database, then ask it to show you one entry:

```
db.got.findOne()
```

You should see a long document, which has three main properties:

- A book title.
- A book chapter number.
- The text, which is a list/array. Each of the entries corresponds to a paragraph, and it is itself a list/array of the sentences that comprise the paragraph.

For some queries this data structure might be useful. But for other queries, we will bring it in a more familiar form, by "unwinding" those lists. $unwind[2] takes a field of the document whose contents is an array, and creates clones of the document, one for each value in the array. It offers you two forms, where on the second form it also creates a new field to keep track of the array index. We use this in the following code, to unwind the nested arrays of paragraphs and sentences into separate documents. This is what this next code does, and it writes the result in a new collection, called got2:

```
db.got.aggregate([
  { $unwind: {
    path: "$text",
    includeArrayIndex: "paragraph"
  }},
  { $unwind: {
    path: "$text",
    includeArrayIndex: "sentence"
  }},
```

[1] http://www.readbooksvampire.com/George_R.R._Martin/
[2] https://docs.mongodb.com/v3.2/reference/operator/aggregation/unwind/#pipe._S_unwind

```
  { $project: {
    _id: false,
    chapter: true,
    title: true,
    paragraph: true,
    sentence: true,
    text: true
  }},
  { $out: "got2" }
])
```

Now run the following to see what these entries look like:

```
db.got2.findOne()
```

You will see a document that corresponds to a sentence. It contains 5 main entries:

- The title and chapter number as before.
- A paragraph number and a sentence number that count starting at 0.
- The actual text for that sentence.

For future use, we will also "index" the text in those documents, with the following:

```
db.got2.createIndex({ text: "text" })
```

Finally we will create one more collection. This collection contains individual word entries. This will also take some time to run:

```
var requests = [];
db.got2.find().forEach(function(doc) {
  requests.push({
    title: doc.title,
    chapter: doc.chapter,
    paragraph: doc.paragraph,
    sentence: doc.sentence,
    words: doc.text.split(/\W+/)
  });
})
db.got3.insertMany(requests)
```

After you have done this, try db.got3.findOne() to see how one entry looks like. You will see that each document corresponds to a sentence with the words in it split into an array.

Here are the questions to answer. Add your answers to this assignment6.js file[3]. The file is not meant to run as is, it's just a place where you put your answers.

1. Find out how many sentences there are on each book. This should be a simple aggregate on the got2 collection, with a $group stage.

[3] https://raw.githubusercontent.com/skiadas/DataWranglingCourse/gh-pages/assignments/assignment6.js

2. Find out how many sentences there are on each paragraph. Order the results by the "longest" paragraph. You can do this with an aggregate on the got2 collection, with a $group stage with a suitable _id entry to group the appropriate entries, followed by a $sort stage.
3. The following line finds all "sentences" with length no more than 3 characters: db.got2.count({ $where: "this.text.length <= 3" }) Change it to instead remove these entries from the got2 collection altogether.
4. Compute the average number of sentences per paragraph for each chapter of each book. Order the results by decreasing average number of sentences per paragraph. You can do this with an aggregate on the got2 collection, consisting of two $group stages, one to compute the number of sentences within each paragraph, and one to average those numbers over all paragraphs of each chapter.
5. Similar to the 4th problem, compute the average number of sentences per paragraph, but now for each book rather than for each chapter separately.
6. Count how many times each word appears. You will need to use an aggregate on got3 with $unwind as a first step. Order the words based on their frequency, starting from the most frequent. By manual inspection, go through the first 30-40 entries and identify at least 5 results that you consider "interesting" in some way, and why.
7. Count how many times each word appears within each book. So the resulting "documents" will contain a title field, and a word field, that together would form the _id, and also a count field that counts the occurences of the word within the titled book. Order the results by descending frequency.
8. (This is more challenging) Continuing from the previous problem, we want to select from each book the 20 most frequently occuring words. To do so, you need to add more stages to your pipeline from the previous problem, to do the following:

    - Use a $group stage with a suitable $push operator in it to create a document for each book, with a field that contains an array of objects, each containing a word and its frequency. See here[4] for an example.
    - Use a $project with a $slice operator to only keep the first 20 entries. See this doc[5] for an example.
    - Use $unwind to expand this array of entries into separate documents. You should end up with 20 entries for each document. See here[6] for use of $unwind.

9. Choose six people from the first five Game of Thrones books. Then using the got2collection find the number of sentences where each person is mentioned. This would be six different queries, or you can do a for-loop. You will want to use the $text operator[7]. Note especially the "exact phrase" syntax, if you want to match a person by both their full name (first and last name). This is not an aggregate query, you will probably want to use a count query, or a find query when you want to look at the answers.

[4] https://docs.mongodb.com/v3.2/reference/operator/aggregation/push/#grp._S_push
[5] https://docs.mongodb.com/v3.2/reference/operator/aggregation/slice/#exp._S_slice
[6] https://docs.mongodb.com/v3.2/reference/operator/aggregation/unwind/#pipe._S_unwind
[7] https://docs.mongodb.com/v3.2/text-search/

When you are ready to submit, make sure the assignment6.js file you have been working on is saved and contains all the answers clearly marked, then email it to me.