Alignment-free approaches to rapidly query massive sequence datasets exploiting NoSQL technologies

FABIO CUMBO

Proposal 1

Document-Based DB

Basics

MongoDB

Alignment-free approaches to rapidly query massive sequence datasets exploiting NoSQL technologies

First approach: Document-Based DB

MongoDB is a
scalable, high-performance, open source,
schema-free, document-oriented database.

mongodb.org

MongoDB is a
scalable, high-performance, open source,
schema-free, document-oriented database

mongodb.org

Non-relational DBs scale more easily,
especially horizontally, through **automatic sharding**

How To

represent experiments in MongoDB

DOCUMENT-BASED

Documents are stored in BSON (binary JSON).
BSON is a binary serialization of JSON-like objects.

Any valid JSON can be easily imported and queried.

REPRESENTING EXPERIMENTS AS JSON DOCUMENTS

Three possible approaches:

1. Store the **whole experiment** in one (or more) JSON document
2. Store all **< K , [E_1 , E_2 , … , E_n] > couples** as JSON documents
    - K is a k-mer
    - E is an experiment ID
    - [E_1 , E_2 , … , E_n] is the set of experiment in which K appears
3. Store all **< E , [K_1 , K_2 , … , K_m] > couples** as JSON documents
    - E is an experiment ID
    - [K_1 , K_2 , … , K_m] is the set of the k-mers in the experiment E

1st APPROACH

Store the whole experiment in one (or more) JSON document

EXPERIMENT SAMPLE

FASTQ Format

```
@EXPERIMENT1_000000001
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
@EXPERIMENT1_000000002
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
@EXPERIMENT1_000000003
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
@EXPERIMENT1_000000004
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
```

1st APPROACH

Store the whole experiment in one (or more) JSON document

EXPERIMENT SAMPLE

FASTQ Format

```
@EXPERIMENT1_000000001
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
@EXPERIMENT1_000000002
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
@EXPERIMENT1_000000003
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
@EXPERIMENT1_000000004
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
```

1st APPROACH

Store the whole experiment in one (or more) JSON document

EXPERIMENT SAMPLE

```
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
```

DOCUMENT REPRESENTATION

JSON Format

```
{
   "experiment": "EXPERIMENT1",
   "source": "Sequence Read Archive",
   "timestamp": "2018-05-17 13:05:42.494039",
   "sequence":
                "CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCG
                 TGGCATTCATGGAGGATGGGAGCCTGCTCACCCTGCATACA
                 CTCCAGGAGTTTCTACATTCTCGGCATCTCCTGATTGGCTG
                 GCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAA
                 AGTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGA
                 TGGCTCAGTACGCGT"
}
```

1st APPROACH

Store the whole experiment in one (or more) JSON document

EXPERIMENT SAMPLE

```
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
```

DOCUMENT REPRESENTATION

JSON Format

```
{
   "experiment": "EXPERIMENT1",
   "source": "Sequence Read Archive",
   "timestamp": "2018-05-17 13:05:42.494039",
   "sequence":
                "CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCG
                 TGGCATTCATGGAGGATGGGAGCCTGCTCACCCTGCATACA
                 CTCCAGGAGTTTCTACATTCTCGGCATCTCCTGATTGGCTG
                 GCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAA
                 AGTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGA
                 TGGCTCAGTACGCGT"
}
```

MONGODB LIMITATION

The document size is limited to **16MB**

1st APPROACH

Store the whole experiment in one (or more) JSON document

EXPERIMENT SAMPLE

```
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
```

STRING COMPRESSION

Try to reduce the length of the sequence

C`TTTTTT`CTCTCT`GGCC`T`CC`TAT`AAAAA` = C`T6`CTCTCT`G2C2`T`C2`TAT`A5`

DOCUMENT REPRESENTATION

JSON Format

```
{
    ...
    "sequence":
                "CT3CTG2TC3AGCA2GATCACAGCT2GCTG2CGTGT2CGT
                 G2CAT2CATG2AG2ATG3AGC2TGCTCAC3TGCATACACT
                 C2AG2AGT3CTACAT2CTCG2CATCTC2TGAT2G2CTG2C
                 AGT2GT3C2T2CT6CTCTCTG2C2TC2TATA5GTCTGCAT
                 A2GC2ACTGAC2TG2CGCG2TG2ATGA2GTGATG2CTCAG
                 TACGCGT"
}
```

1st APPROACH

Store the whole experiment in one (or more) JSON document

EXPERIMENT SAMPLE

```
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
```

STRING COMPRESSION

Try to reduce the length of the sequence

```
CTTTTTTCTCTCTGGCCTCCTATAAAAA = CT6CTCTCTG2C2TC2TATA5
```

DOCUMENT REPRESENTATION

JSON Format

```
{
    ...
    "sequence":
                "CT3CTG2TC3AGCA2GATCACAGCT2GCTG2CGTGT2CGT
                 G2CAT2CATG2AG2ATG3AGC2TGCTCAC3TGCATACACT
                 C2AG2AGT3CTACAT2CTCG2CATCTC2TGAT2G2CTG2C
                 AGT2GT3C2T2CT6CTCTCTG2C2TC2TATA5GTCTGCAT
                 A2GC2ACTGAC2TG2CGCG2TG2ATGA2GTGATG2CTCAG
                 TACGCGT"
}
```

COULD BE NOT ENOUGH FOR LONG SEQUENCES

1st APPROACH

Store the whole experiment in one (or more) JSON document

EXPERIMENT SAMPLE

```
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
```

DOCUMENT REPRESENTATION

JSON Format

SPLIT THE SEQUENCE

Split the sequence in multiple chunks and take trace of the chunk number to eventually rebuild the original sequence

Limit the number of characters per sequence to be sure that the BSON size will not be greater than MongoDB Document size limit (16MB)

```
{
    ...
    "sequence": "CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTC",
    "part": 0
}

{
    ...
    "sequence": "CACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAGGATGG",
    "part": 1
}
...
```

2nd APPROACH

Store all < K , [E_1 , E_2 , … , E_n] > couples as JSON documents

- K is a k-mer
- E is an experiment ID
- [E_1 , E_2 , … , E_n] is the set of experiment in which K appears

3rd APPROACH

Store all < E , [K_1 , K_2 , … , K_m] > couples as JSON documents

- E is an experiment ID
- [K_1 , K_2 , … , K_m] is the set of the k-mers in the experiment E

EXPERIMENT SAMPLE

```
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
```

K-MERS EXTRACTION

Extract all the possible substring of length K that are contained in the sequence

JELLYFISH K-MER COUNTER

K = 21

STRING COMPRESSION

```
AAAAAAAAAAAAAAAAAAAAA
GTTACTGAGCCATCTGCTGGA
AAAACTCTAATAAATCAGGCT
GGCAAAGCCCCCAAGGTCACC
GGACCCTCCCAATGGGGAAAA
ATTTCCAGTACTATGATGTCA
...
```

```
A21
GT2ACTGAGC2ATCTGCTG2A
A4CTCTA2TA3TCAG2CT
G2CA3GC5A2G2TCAC2
G2AC3TC3A2TG4A4
AT3C2AGTACTATGATGTCA
...
```

2nd APPROACH

Store all < K , [E_1 , E_2 , … , E_n] > couples as JSON documents

- K is a k-mer
- E is an experiment ID
- [E_1 , E_2 , … , E_n] is the set of experiment in which K appears

DOCUMENT REPRESENTATION

JSON Format

```
{
   "kmer": "A21",
   "experiments": [ "EXPERIMENT1", "EXPERIMENT2", ... ]
}

{
   "kmer": "GT2ACTGAGC2ATCTGCTG2A"
   "experiments": [ "EXPERIMENT13", "EXPERIMENT87", ... ]
}

{
   "kmer": "A4CTCTA2TA3TCAG2CT"
   "experiments": [ "EXPERIMENT13", "EXPERIMENT87", ... ]
}

...
```

3rd APPROACH

Store all < E , [K_1 , K_2 , … , K_m] > couples as JSON documents

- E is an experiment ID
- [K_1 , K_2 , … , K_m] is the set of the k-mers in the experiment E

DOCUMENT REPRESENTATION

JSON Format

```
{
   "experiment": "EXPERIMENT1",
   "kmers": [
              "A21",
              "GT2ACTGAGC2ATCTGCTG2A",
              "A4CTCTA2TA3TCAG2CT",
              "G2CA3GC5A2G2TCAC2",
              "G2AC3TC3A2TG4A4",
              "AT3C2AGTACTATGATGTCA",
              ...
            ]
}

...
```

MONGODB LIMITATION … AGAIN

The document size is limited to **16MB**

2nd APPROACH

DOCUMENT REPRESENTATION

JSON Format

Limit the size of the "experiments" array to be sure that the BSON size will not be greater than MongoDB Document size limit (**16MB**)

```
{
   "kmer": "A21",
   "experiments": [ "EXPERIMENT1", "EXPERIMENT2", ... ],
   "part": 0
},
{
   "kmer": "A21"
   "experiments": [ "EXPERIMENT23", "EXPERIMENT37", ... ],
   "part": 1
},
{
   "kmer": "A21"
   "experiments": [ "EXPERIMENT74", "EXPERIMENT91", ... ],
   "part": 2
},
...
```

MONGODB LIMITATION … AGAIN

The document size is limited to **16MB**

3rd APPROACH

DOCUMENT REPRESENTATION

JSON Format

Limit the size of the "kmers" array to be sure that the BSON size will not be greater than MongoDB Document size limit (**16MB**)

```
{
   "experiment": "EXPERIMENT1",
   "kmers": [ "A21", "GT2ACTGAGC2ATCTGCTG2A", ... ],
   "part": 0
},
{
   "experiment": "EXPERIMENT1"
   "kmers": [ "A4CTCTA2TA3TCAG4", "G2CA3GC5A2G2TC3", ... ],
   "part": 1
},
{
   "experiment": "EXPERIMENT1"
   "kmers": [ "G2AC3TC3A2TG4A4", "AT3C2AGTA7TGTCA", ... ],
   "part": 2
},
...
```

How To

query MongoDB

PROBLEM Retrieve all the experiments which express a particular transcript.

QUERY SAMPLE

```
ACAGTGCTAGATTTTTCACATTTTTTCTCACTTCTTTTCATATTTAAAA
```

EXTRACT K-MERS
K = 21

```
[ "ACAGTGCTAGATTTTTCACAT", "TTTTTCACATTTTTTCTCACT",
  "TCACTTCTTTTCATATTTAAA", ... ]
```

COMPRESS K-MERS

```
[ "ACAGTGCTAGAT5CACAT", "T5CACAT6CTCACT",
  "TCACT2CT4CATAT3A3", ... ]
```

FOR EACH K-MER QUERY MONGODB

1st APPROACH

```
db.collection.find( { sequence: { $regex: /T5CACAT6CTCACT/ } },
                    { experiment: 1 } )
```

2nd APPROACH

```
db.collection.find( { kmer: "T5CACAT6CTCACT" },
                    { experiments: 1 } )
```

3rd APPROACH

```
db.collection.find( { kmers: "T5CACAT6CTCACT" },
                    { experiment: 1 } )
```

Comparing
the three approaches

	1st APPROACH	2nd APPROACH	3rd APPROACH
DB VOLUME	★★☆	★★★	★☆☆
DB CONSTRUCTION TIME	★★★ No need to extract all the k-mers	★☆☆ Need to potentially store billions of documents (`4^21` k-mers). **Is it just a theoretical limitation?**	★★☆
DB QUERY TIME	★☆☆ Using regular expressions to find k-mers would make the queries **extremely slow.** This approach is not feasible.	★★★ Creating an index on the k-mer field will drastically accelerate the query time.	★★☆ Creating an index on the k-mers fields would accelerate the full-text search (perfect string matching) but it requires to search on all documents.

Improvements

Some proposals

PROBLEM

Create indices would be good to reduce the search time, but using one gigantic collection (with billions (or more) documents) could not be enough.

POSSIBLE SOLUTION

Create multiple collections and generate and index for each of them.

"Generally, having a large number of collections has no significant performance penalty, and results in very good performance"
(from MongoDB Documentation).

The default maximum number of namespaces is 24K, it means that MongoDB can handle at most 24K collections but it can be increased up to 3M.

HOW TO ORGANIZE DATA

- How many collections?
- Following what kind of logic?

HOW TO ORGANIZE DATA

Splitting the data by **`50%`**, or **`25%`**, or in general by **`N%`**, will produce **`N`** collections. This approach obviously does not work because, during the query process, MongoDB has to search in all the **`N`** collections.

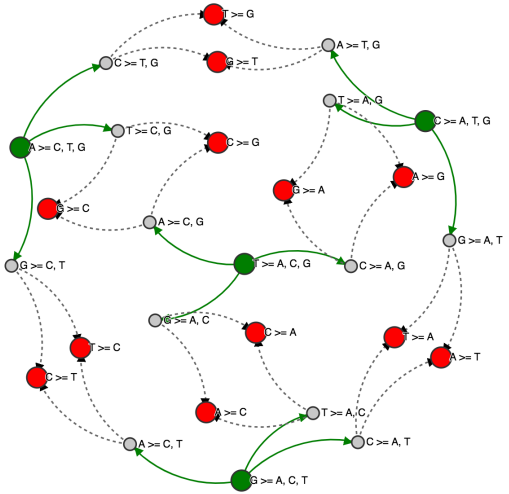
Is there some logic to intelligently split the data?

FREQUENCY-BASED DECISION GRAPH

We are collecting compressed k-mers like "`A4CTCTA2TA3TCAG4`". Thinking at the frequency of the nucleotides in the k-mer, we could establish how to categorize k-mers.

As in the case of finite-state machines:

- green nodes represent source states;
- grey nodes are intermediate states;
- red nodes are accepting states



EXTRACTING LOGIC FORMULAS

Following all the paths in the Frequency-Based Decision Graph, we could extract the following **24** logic formulas:

F1. (f(A)>=f(C) **OR** f(A)>=f(T) **OR** f(A)>=f(G)) **AND** (f(C)>=f(T) **OR** f(C)>=f(G)) **AND** (f(T)>=f(G))
F2. (f(A)>=f(C) **OR** f(A)>=f(T) **OR** f(A)>=f(G)) **AND** (f(C)>=f(T) **OR** f(C)>=f(G)) **AND** (f(G)>=f(T))
F3. (f(A)>=f(C) **OR** f(A)>=f(T) **OR** f(A)>=f(G)) **AND** (f(T)>=f(C) **OR** f(T)>=f(G)) **AND** (f(C)>=f(G))
F4. (f(A)>=f(C) **OR** f(A)>=f(T) **OR** f(A)>=f(G)) **AND** (f(T)>=f(C) **OR** f(T)>=f(G)) **AND** (f(G)>=f(C))
F5. (f(A)>=f(C) **OR** f(A)>=f(T) **OR** f(A)>=f(G)) **AND** (f(G)>=f(C) **OR** f(G)>=f(T)) **AND** (f(C)>=f(T))
F6. (f(A)>=f(C) **OR** f(A)>=f(T) **OR** f(A)>=f(G)) **AND** (f(G)>=f(C) **OR** f(G)>=f(T)) **AND** (f(T)>=f(C))

F7. (f(C)>=f(A) **OR** f(C)>=f(T) **OR** f(C)>=f(G)) **AND** (f(A)>=f(T) **OR** f(A)>=f(G)) **AND** (f(T)>=f(G))
F8. (f(C)>=f(A) **OR** f(C)>=f(T) **OR** f(C)>=f(G)) **AND** (f(A)>=f(T) **OR** f(A)>=f(G)) **AND** (f(G)>=f(T))
F9. (f(C)>=f(A) **OR** f(C)>=f(T) **OR** f(C)>=f(G)) **AND** (f(T)>=f(A) **OR** f(T)>=f(G)) **AND** (f(A)>=f(G))
F10. (f(C)>=f(A) **OR** f(C)>=f(T) **OR** f(C)>=f(G)) **AND** (f(T)>=f(A) **OR** f(T)>=f(G)) **AND** (f(G)>=f(A))
F11. (f(C)>=f(A) **OR** f(C)>=f(T) **OR** f(C)>=f(G)) **AND** (f(G)>=f(A) **OR** f(G)>=f(T)) **AND** (f(A)>=f(T))
F12. (f(C)>=f(A) **OR** f(C)>=f(T) **OR** f(C)>=f(G)) **AND** (f(G)>=f(A) **OR** f(G)>=f(T)) **AND** (f(T)>=f(A))

F13. (f(T)>=f(A) **OR** f(T)>=f(C) **OR** f(T)>=f(G)) **AND** (f(A)>=f(C) **OR** f(A)>=f(G)) **AND** (f(C)>=f(G))
F14. (f(T)>=f(A) **OR** f(T)>=f(C) **OR** f(T)>=f(G)) **AND** (f(A)>=f(C) **OR** f(A)>=f(G)) **AND** (f(G)>=f(C))
F15. (f(T)>=f(A) **OR** f(T)>=f(C) **OR** f(T)>=f(G)) **AND** (f(C)>=f(A) **OR** f(C)>=f(G)) **AND** (f(A)>=f(G))
F16. (f(T)>=f(A) **OR** f(T)>=f(C) **OR** f(T)>=f(G)) **AND** (f(C)>=f(A) **OR** f(C)>=f(G)) **AND** (f(G)>=f(A))
F17. (f(T)>=f(A) **OR** f(T)>=f(C) **OR** f(T)>=f(G)) **AND** (f(G)>=f(A) **OR** f(G)>=f(C)) **AND** (f(A)>=f(C))
F18. (f(T)>=f(A) **OR** f(T)>=f(C) **OR** f(T)>=f(G)) **AND** (f(G)>=f(A) **OR** f(G)>=f(C)) **AND** (f(C)>=f(A))

F19. (f(G)>=f(A) **OR** f(G)>=f(C) **OR** f(G)>=f(T)) **AND** (f(A)>=f(C) **OR** f(A)>=f(T)) **AND** (f(C)>=f(T))
F20. (f(G)>=f(A) **OR** f(G)>=f(C) **OR** f(G)>=f(T)) **AND** (f(A)>=f(C) **OR** f(A)>=f(T)) **AND** (f(T)>=f(C))
F21. (f(G)>=f(A) **OR** f(G)>=f(C) **OR** f(G)>=f(T)) **AND** (f(C)>=f(A) **OR** f(C)>=f(T)) **AND** (f(A)>=f(T))
F22. (f(G)>=f(A) **OR** f(G)>=f(C) **OR** f(G)>=f(T)) **AND** (f(C)>=f(A) **OR** f(C)>=f(T)) **AND** (f(T)>=f(A))
F23. (f(G)>=f(A) **OR** f(G)>=f(C) **OR** f(G)>=f(T)) **AND** (f(T)>=f(A) **OR** f(T)>=f(C)) **AND** (f(A)>=f(C))
F24. (f(G)>=f(A) **OR** f(G)>=f(C) **OR** f(G)>=f(T)) **AND** (f(T)>=f(A) **OR** f(T)>=f(C)) **AND** (f(C)>=f(A))

LOGIC FORMULAS

24 logic formulas means that we can intelligently split the data in **24** different collections.

Could we do better?

SPLITTING DATA ACCORDING TO THE COMPRESSED STRING LENGTH

We are collecting compressed k-mers like:

```
A21
G2AC3TC3A2TG4A4
G2CA3GC5A2G2TCAC2
A4CTCTA2TA3TCAG2CT
GT2ACTGAGC2ATCTGCTG2A
```

We could additionally split the data according to the compressed string length, resulting in **21** additional collections after every single logic formula because of the chosen **`K = 21`**.

In the case of this kind of compressed strings, le shortest one has a length equals to `3` (e.g. `A21`), decreasing the number of additional collections per formula to `19`, resulting in `( 24 * 19 ) = 456` collections.

CURRENT SITUATION

24 logic formulas, with **19** collections each, resulting in **(24 * 19) = 456** total collections.

Could we do better?

HOW TO ASSIGN A SCORE TO A K-MER

The goal is to assign a numerical score (in the interval **[0, 1]**) to a given string (compressed k-mer). A simple solution would consist on replacing the nucleotides with the corresponding position in the English alphabet (starting from **0** to **25**) and sum the resulting numbers:

```
kmer_sample = G2AC3TC3A2TG4A4
[ G, 2, A, C, 3, T,  C, 3, A, 2, T,  G, 4, A, 4 ]
[ 6, 2, 0, 2, 3, 19, 2, 3, 0, 2, 19, 6, 4, 0, 4 ]
partial score = 72
```

The sum of every element in the numeric array should be divided by the maximum admissible partial score (**269**), represented by the k-mer `TGTGTGTGTGTGTGTGTGTGT` (considering **K = 21**).

The final score for the `kmer_sample` is `0,2676579926`.

Considering only the first two decimals, it will result in **100** collections.

THEORETICAL NUMBER OF COLLECTIONS

Considering the previously extracted **24 logic formulas**, the **19 additional collections** based on the compressed k-mers length, and the **100 additional collections** based on the string score, it will result in the theoretical total amount of `( 24 * 19 * 100 ) = 45,600` collections.

HOW TO QUERY THE DB

1. Query sample:

`ACAGTGCTAGATTTTTCACATTTTTTCTCACTTCTTTTCATATTTAAAA`

2. Extract all compressed k-mers:

`[ ACAGTGCTAGAT5CACAT, T5CACAT6CTCACT, T6CTCACT2CT4CAT, ... ]`

3. For each compressed k-mer:

`Current k-mer: ACAGTGCTAGAT5CACAT`

   1. Apply logic formulas to identify the first set of collections :

   `Verified formulas: [ F1, F3, F7, F9, F13, F15 ]`

   2. Consider the length of the current k-mer, for both the set of collections representing by the previously selected formulas:

   `k-mer length: 18`

   `Candidate goups of collections:`

   `[ F1.L18, F3.L18, F7.L18, F9.L18, F13.L18, F15.L18 ]`

   3. Assign a score to the k-mer

   `k-mer score: ( 107 / 269 ) ~ 0,39`

   `Search in: [ F1.L18.S039, F3.L18.S039, F7.L18.S039, F9.L18.S039, F13.L18.S039, F15.L18.S039 ]`

THEORETICAL NUMBER OF COLLECTIONS

Considering the previously extracted **24 logic formulas**, the **19 additional collections** based on the compressed k-mers length, and the **100 additional collections** based on the string score, it will result in the theoretical total amount of `( 24 * 19 * 100 ) = 45,600` collections.

HOW TO QUERY THE DB

1. Query sample:

`ACAGTGCTAGATTTTTCACATTTTTTCTCACTTCTTTTCATATTTAAAA`

2. Extract all compressed k-mers:

`[ ACAGTGCTAGAT5CACAT, T5CACAT6CTCACT, T6CTCACT2CT4CAT, ... ]`

3. For each compressed k-mer:

`Current k-mer: ACAGTGCTAGAT5CACAT`

   1. Apply logic formulas to identify the first set of collections :

   `Verified formulas: [ F1, F3, F7, F9, F13, F15 ]`

   2. Consider the length of the current k-mer, for both the set of collections representing by the previously selected formulas:

   `k-mer length: 18`

   `Candidate goups of collections:`

   `[ F1.L18, F3.L18, F7.L18, F9.L18, F13.L18, F15.L18 ]`

   3. Assign a score to the k-mer

   `k-mer score: ( 107 / 269 ) ~ 0,39`

   `Search in: [ F1.L18.S039, F3.L18.S039, F7.L18.S039, F9.L18.S039, F13.L18.S039, F15.L18.S039 ]`

WITH REGARD TO THIS EXAMPLE, WE HAVE TO SEARCH THE K-MER IN 6 COLLECTIONS ONLY

CONSIDERATIONS

In the previous example, for the k-mer `ACAGTGCTAGAT5CACAT`, **6** logic formulas has been verified. It means that, if that k-mer was previously inserted in the DB, it exists for sure in **6** collections (**data redundancy**), so **we could limit the search of the k-mer to one of this collections only.**

QUESTION

Is it possible to reduce the data redundancy rate to make sure that a k-mer exists in one and only one collection?

POSSIBLE SOLUTION

Consider the example on the previous slide. The following logic formulas have been verified:

```
[ F1, F15, F9, F13, F7, F3 ]
```

Merging the selected formulas will result in a single combined representation of the candidate collection (after the formula IDs have been sorted):

```
[ F1_F3_F7_F9_F13_F15 ]
```

This solution allow us to **completely remove the data redundancy** but it will **drastically increase the number of collections** in the DB **(combinatorial explosion)**. **A generic k-mer could verify `1` up to all the `24` logic formulas.**

LET'S COUNT

PYTHON SNIPPET

```
from itertools import combinations
FORMULAS = range(1, 25) # 25 is not included in the range
COLLECTIONS = 0
for comb_size in range(1, 25):
    COLLECTIONS += len( [ ",".join(map(str, comb)) for comb in combinations(FORMULAS, comb_size) ] )
```

LET'S COUNT

PYTHON SNIPPET

```
from itertools import combinations
FORMULAS = range(1, 25) # 25 is not included in the range
COLLECTIONS = 0
for comb_size in range(1, 25):
    COLLECTIONS += len( [ ",".join(map(str, comb)) for comb in combinations(FORMULAS, comb_size) ] )
```

~17M COLLECTIONS

JUST A THEORETICAL LIMIT

PARTITIONING ON K-MER LENGTH AND
K-MER SCORE IS NO MORE REQUIRED

CONSIDERATIONS

The previous approaches aim to partition the k-mers in multiple collections.
The previously described techniques will produce unbalanced collections, some of them with a lot of documents, and other with a few documents.

HASH FUNCTION

Using a hash function to establish where to put a document will produce a homogeneous distribution of the documents in the set of collections.

Of course, the number of collection has to be decided a priori.

```
import xxhash
from itertools import product

ALPHABET = “ACGT”
COLLECTIONS = 10000
KMER_LEN = 21
KMERS = [ "".join(map(str, prod)) for prod in product( ALPHABET, repeat=KMER_LEN ) ]
HASH_SEED = 1234

for kmer in KMERS:
    collection = ( int(xxhash.xxh64(kmer, seed=HASH_SEED).intdigest()) % COLLECTIONS )
    ...
```

CONSIDERATIONS

Generating a huge number of collections is a good idea. However, sometimes the number of documents could be in the order of billions or more.

QUESTION

Is it possible to reduce the number of documents?

IDEA

Could we make use of minimizers?

MINIMUM SUBSTRING PARTITIONING

It comes from the intuition that two adjacent k-mers are very likely to share the same minimum p-substring if **`p << k`**, since there is a length-(**`k-1`**) overlap between them.

MINIMIZERS

Using the minimizers will produce a lot of advantages. It will drastically reduce the number of documents and consequently the number of collections (the theoretical maximum amount of collections will be `4^p` with `p` usually equals to `6`, producing `4096` collections, one for each minimizer).

`4096` is not a huge number, so the best choice would be considering one collection only that will contain at most `4096` documents, one for each minimizer

LET'S TRY SOME MACHINE LEARNING

Considering a **set of organisms** and a **set of experiments for each organism** already stored in MongoDB, we can build a **classification model** that will be able to **establish if a given experiment is related to one or more organisms, reporting the accuracy percentage for each of them.**
The dataset could be represented as a matrix like the following one:

	A3CG2	AC4T	C2G4	TCTA3	...	T4GC	class
EXP1	1	1	0	1		1	ORGANISM1
EXP2	0	1	0	0		1	ORGANISM1
EXP3	0	0	1	1		0	ORGANISM2
EXP4	1	1	1	1		1	ORGANISM2
...							
EXPn	1	0	0	1		1	ORGANISMn

HOW TO INTEGRATE A CLASSIFIER IN MONGODB WITHOUT THE NEED TO EXTRACT THE DATA?

Proposal 2

Graph DB

Basics

Neo4j

Neo4j is a graph database management system with native graph storage and processing.

Everything is stored in the form of either an edge, a node, or an attribute.
Each node and edge can have any number of attributes.
Both the nodes and edges can be labelled.

neo4j.com

How To

represent experiments in neo4j

DE BRUIJN GRAPH

Represent a sequence in terms of its k-mer components

1. Choose a k-mer size and split the original sequence into its k-mer components
2. Build a directed graph by connecting pairs of k-mers with overlaps between the first `k-1` nucleotides and the last `k-1` nucleotides
3. The direction of the edge goes from the k-mer whose last `k-1` nucleotides are overlapping, to the k-mer whose the first `k-1` nucleotides are overlapping

EXPERIMENT SAMPLE

```
CTTTCTGGTCCCAGCAAGATCACAGCTTGCTGGCGTGTTCGTGGCATTCATGGAG
GATGGGAGCCTGCTCACCCTGCATACACTCCAGGAGTTTCTACATTCTCGGCATC
TCCTGATTGGCTGGCAGTTGTTTCCTTCTTTTTTCTCTCTGGCCTCCTATAAAAA
GTCTGCATAAGCCACTGACCTGGCGCGGTGGATGAAGTGATGGCTCAGTACGCGT
```

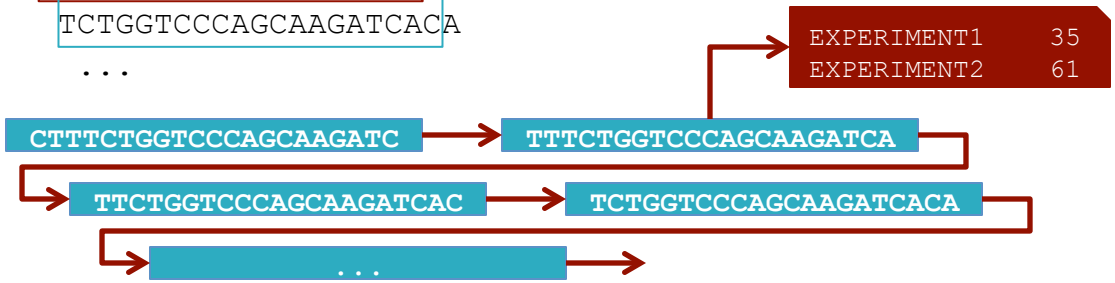
K-MERS

`K = 21`

```
CTTTCTGGTCCCAGCAAGATC
 TTTCTGGTCCCAGCAAGATCA
  TTCTGGTCCCAGCAAGATCAC
   TCTGGTCCCAGCAAGATCACA
    . . .
```

DE BRUIJN GRAPH REPRESENTATION

For each node take trace of the frequency of the k-mer for each experiment in which the k-mer occurs on



Alignment-free approaches to rapidly query massive sequence datasets exploiting NoSQL technologies

First approach: Graph DB

Graph ML

GRAPH MACHINE LEARNING

1. It allows the learning system to explore more of your data
2. It can analyze the relationships between entities as well as their properties

KERAS

High level wrapper around deep learning frameworks like Tensorflow.
"Keras lets you build and train a model with just a few lines of code, as the library takes care of much of the repetitive boilerplate".

IDEA

1. Create a De Bruijn graph representing a set of experiments (for multiple organisms)
2. Annotate every node with information like the frequency of the k-mer (represented by the node) in all the examined experiments
3. The result will be a De Bruijn graph that represents the genomes of the considered organisms (all in one graph – it will be really dense)
4. Create a classification model using tools like Keras

GOAL

5. **Apply the generated model to establish if a given experiment could be related to one or more (in percentage – accuracy) organisms**