Using Elasticsearch to **store**, **integrate** and **mine** diverse biological data

Andrea Pierleoni

CTTV Core Team - EBI

goo.gl/hmTI6j

4 Feb 2016

Centre for Therapeutic Target Validation

CTTV

CTTV

- Private-Public Initiative
  - EBI - Sanger - GSK
- Aims to provide evidence on the biological validity of therapeutic targets using genome-scale experiments and analysis
- CTTV Core
  - The Core Team is in charge of organise and integrate the available data and serve it through a Web App and a REST API
  - Our software stack should be easy to replicate in internal mirrors by partner organisations

Overview

- Data
- Data Integration
- Data Mining
- Architecture

Data

Core Data

Core Data

Data Sources

- ChEMBL
- COSMIC
- Europe PMC
- EVA
- Expression Atlas
- GWAs Catalog
- Phenodigm
- Uniprot
- ... and eventually more
  in private deployments

Core Data

Data Sources

- ChEMBL
- COSMIC
- Europe PMC
- EVA
- Expression Atlas
- GWAs Catalog
- Phenodigm
- Uniprot
- ... and eventually more in private deployments

Data Types

- Genetic association
- Somatic mutation
- RNA expression
- Known drugs
- Animal models
- Text mining

Public Database Data

Public Database Data

- Target related generic data
- Disease related data (EFO Ontology)
- Evidence Ontology
- Reactome pathways
- Protein and RNA expression data
- more to come...

CTTV Building Blocks

TARGET

CTTV Building Blocks

Relate a Target and a Disease through an Evidence

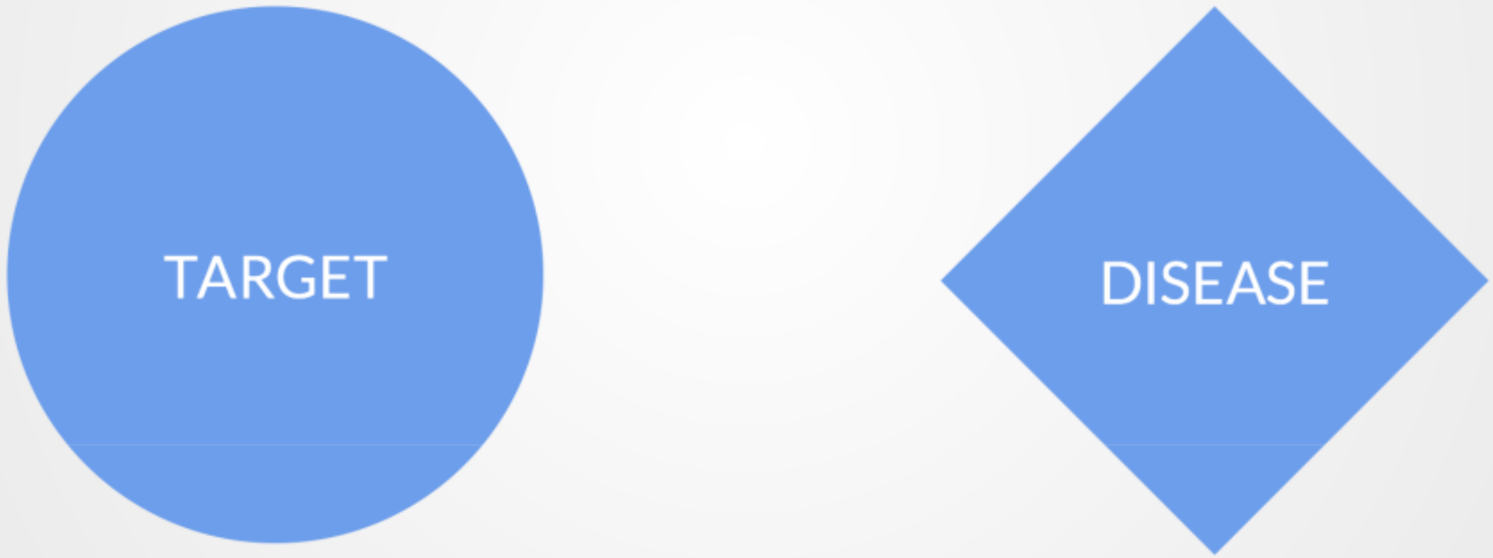
TARGET

CTTV Building Blocks

Relate a Target and a Disease through an Evidence

CTTV Building Blocks

Relate a Target and a Disease through an Evidence



CTTV Building Blocks

Relate a Target and a Disease through an Evidence

CTTV Building Blocks

Relate a Target and a Disease through an Evidence

Building blocks in real life

Building blocks in real life

JSON objects with a flexible format: Evidence String

Building blocks in real life

JSON objects with a flexible format: Evidence String


```
{
    "sourceID": "expression_atlas",
    "type": "rna_expression",
    "target": {
        "id": [
            "http://identifiers.org/ensembl/ENSG00000112984"
        ],
...
    },
    "disease": {
        "id": [
            "http://www.ebi.ac.uk/efo/EFO_0002917"
        ],
...
    },
    "evidence": {
        "unique_experiment_reference": "STUDYID_E-GEOD-14001",
        "resource_score": {
                "value": 0.0168,
                "type":"pvalue"
        },
        "evidence_codes": [
            "http://purl.obolibrary.org/obo/ECO_0000356"
        ],
        "log2_fold_change": {
```

4.2 Millions Evidence Strings

Storing Evidence Strings in Elasticsearch

Storing Evidence Strings in Elasticsearch

PUT JSON into ES

EASY. Just be sure to use a document type per datasource to avoid automatic mapping clashes

Storing Evidence Strings in Elasticsearch

PUT JSON into ES

EASY. Just be sure to use a document type per datasource
to avoid automatic mapping clashes

Filter for Target, Disease and other metadata

Needs ad-hoc mapping

Storing Evidence Strings in Elasticsearch

PUT JSON into ES

EASY. Just be sure to use a document type per datasource
to avoid automatic mapping clashes

Filter for Target, Disease and other metadata

Needs ad-hoc mapping

How many Indexes?

Document count per source range from 20K to 3.3M

Best configuration with a separate index for text mining

Storing Evidence Strings in Elasticsearch

PUT JSON into ES

EASY. Just be sure to use a document type per datasource
to avoid automatic mapping clashes

Filter for Target, Disease and other metadata

Needs ad-hoc mapping

How many Indexes?

Document count per source range from 20K to 3.3M

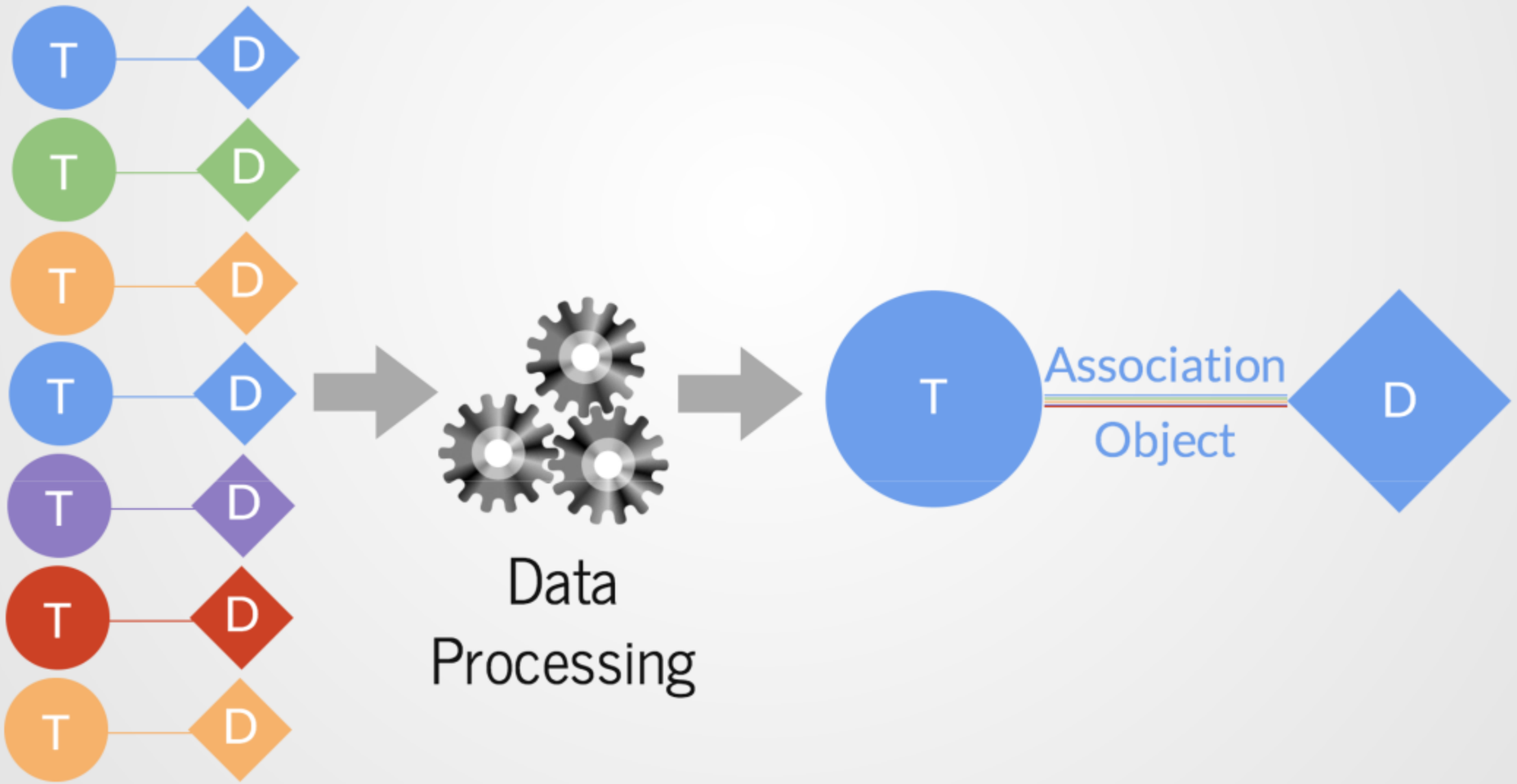
Best configuration with a separate index for text mining

Routing

Routing by target ID distribute nicely the data and
improves filter by target query performances

Data Integration

From single evidence to target-disease association



Data Processing

Strategy #1: compute in Python-based REST API

- Retrieve all relevant evidence strings from ES
- Process them in the api
- Return computed object as JSON

Data Processing

Strategy #1: compute in Python-based REST API

- Retrieve all relevant evidence strings from ES
- Process them in the api
- Return computed object as JSON

PROS

- Easy to code, and experiment with

Data Processing

Strategy #1: compute in Python-based REST API

- Retrieve all relevant evidence strings from ES
- Process them in the api
- Return computed object as JSON

PROS

- Easy to code, and experiment with

CONS

- Very slow to retrieve a big number of evidence strings trough the network
- Python is not the fastest language out there
- Heaviest query took 20 min to run!!

Data Processing

Strategy #2: compute with ES scripted metric aggregation

- Compute heavy processing steps in ES
- Process results the api
- Return computed object as JSON

Data Processing

Strategy #2: compute with ES scripted metric aggregation

- Compute heavy processing steps in ES
- Process results the api
- Return computed object as JSON

PROS

- Much faster, more than 10X over first attempt

Data Processing

Strategy #2: compute with ES scripted metric aggregation

- Compute heavy processing steps in ES
- Process results the api
- Return computed object as JSON

PROS

- Much faster, more than 10X over first attempt

CONS

- Tricky to code and debug
- Heavy load in ES cluster as it requires lots of CPU
- Cannot compute aggregations

Data Processing

Strategy #3: precompute association JSON objects

- Precompute JSON Object in a Python-based pipeline
- Filter and return computed object from the Python REST API

Data Processing

Strategy #3: precompute association JSON objects

- Precompute JSON Object in a Python-based pipeline
- Filter and return computed object from the Python REST API

PROS

- Faster, more than 100X over first attempt.
- All queries have a similar weight on the backend
- Can compute aggregations

Data Processing

Strategy #3: precompute association JSON objects

- Precompute JSON Object in a Python-based pipeline
- Filter and return computed object from the Python REST API

PROS

- Faster, more than 100X over first attempt.
- All queries have a similar weight on the backend
- Can compute aggregations

CONS

- Not able to tune computation parameters in the live system

2.1 Millions Association Objects

Enabling Full Text Search

- Map Association objects to full text search entities
  - Targets
  - Diseases
  - more to come (SNPs, Publications, Drugs, etc...)
- 70K search entities so far (Ensembl Gene Ids + EFO)

Data Processing
Computing Pipeline

Data Processing
Computing Pipeline

Postgres 9.4

CTTV Computing Pipeline

Multi step

Input validation

Evidence String Processing

Association Object Precomputing

Search Object Precomputation

Generic Data Processing

Parallel Processing

Elasticsearch
1.7.2

Data Processing
Computing Pipeline

Postgres 9.4

CTTV Computing Pipeline

Multi step

Input validation

Evidence String Processing

Association Object Precomputing

Search Object Precomputation

Generic Data Processing

Parallel Processing

Elasticsearch
1.7.2

Data Processing
Computing Pipeline

Postgres 9.4

Association Precomputing step:
2 days

CTTV Computing Pipeline

Multi step

Input validation

Evidence String Processing

Association Object Precomputing

Search Object Precomputation

Generic Data Processing

Parallel Processing

Elasticsearch
1.7.2

Data Processing
Computing Pipeline

Postgres 9.4

Association Precomputing step:
2 days

CTTV Computing Pipeline

Multi step

Input validation

Evidence String Processing

Association Object Precomputing

Search Object Precomputation

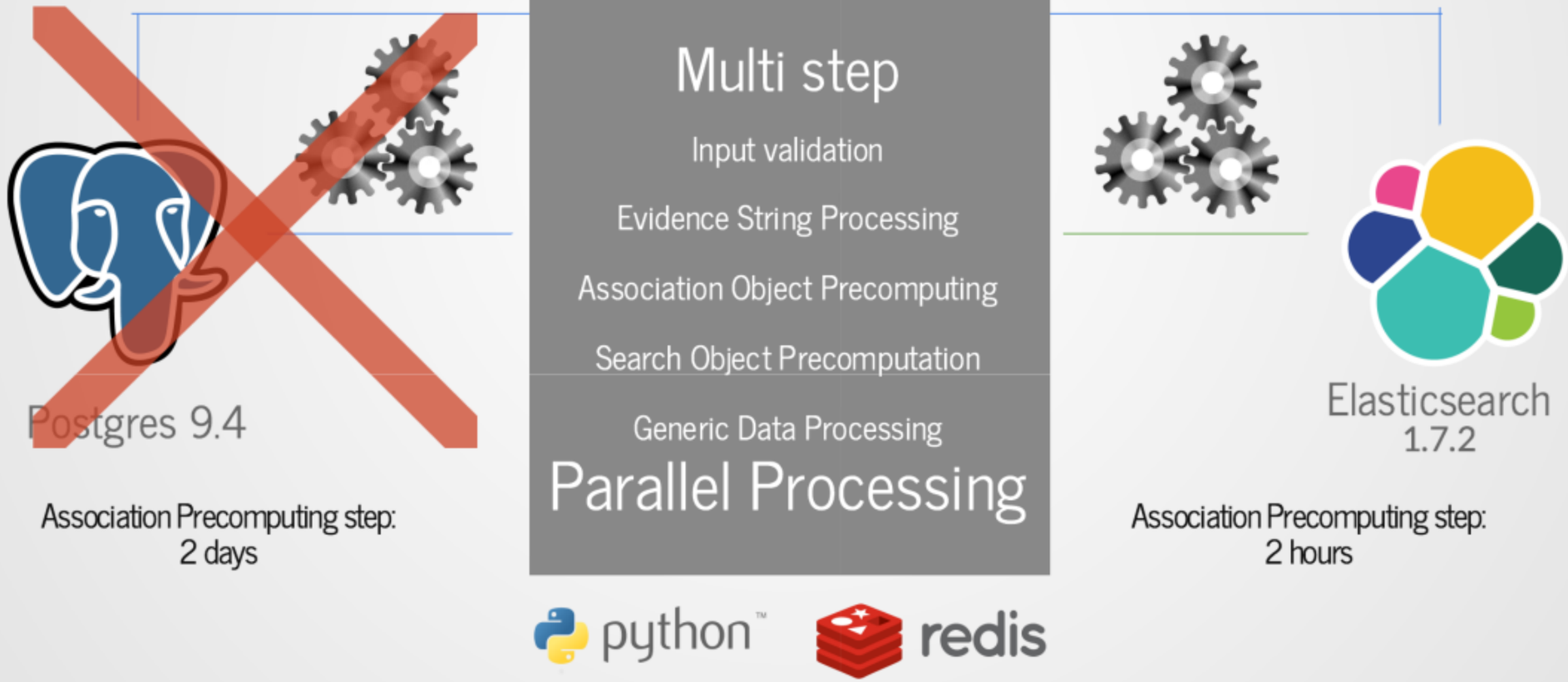
Generic Data Processing

Parallel Processing

Elasticsearch
1.7.2

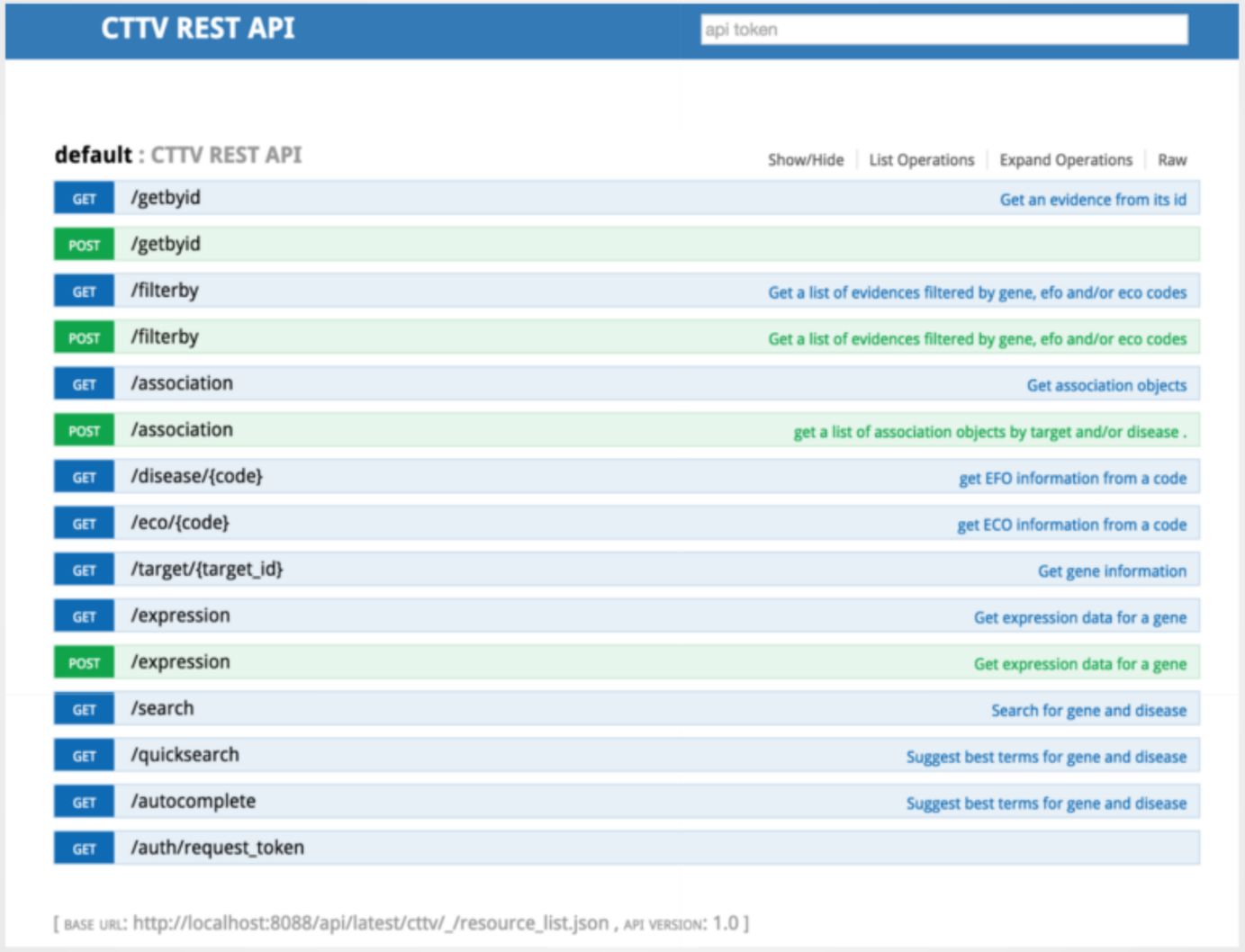
Association Precomputing step:
2 hours

Data Processing
Computing Pipeline



Mining the Data

CTTV REST API



Not public yet (coming soon...)

REST API Implementation

Enable to search for target and disease by full text search
(more options coming)

Most methods are thin wrappers around elasticsearch filters

Returns aggregations for associations to enable faceted search

Auth built in

Full Text Search


⚠WARNING⚠
Demo Time

CTTV Web APP

www.targetvalidation.org

Full Text Search

Full Text Search

Full Text Search

Full Text Search

- Support autocompletion

Full Text Search

- Support autocompletion
- Disambiguate between different document types
  - interleukin
    - TARGET: interleukin-1, interleukin-2, ...
    - DISEASE: Immunodeficiency due to interleukin-1 receptor-associated kinase-4 deficiency

Full Text Search

- Support autocompletion
- Disambiguate between different document types
  - interleukin
    - TARGET: interleukin-1, interleukin-2, ...
    - DISEASE: Immunodeficiency due to interleukin-1 receptor-associated kinase-4 deficiency
- Support for acronymns and many, many target synonyms
  - query: "SLE"
    - SLEep? (EFO term)
    - SLEED1? (Target)
    - systemic lupus erythematosus (what they were actually looking for)

Full Text Search

- Support autocompletion
- Disambiguate between different document types
  - interleukin
    - TARGET: interleukin-1, interleukin-2, ...
    - DISEASE: Immunodeficiency due to interleukin-1 receptor-associated kinase-4 deficiency
- Support for acronymns and many, many target synonyms
  - query: "SLE"
    - SLEep? (EFO term)
    - SLEED1? (Target)
    - systemic lupus erythematosus (what they were actually looking for)
- Data driven search score
  - Increase with the number of associations linked
  - Decrease if down in the ontology tree

Full Text Search

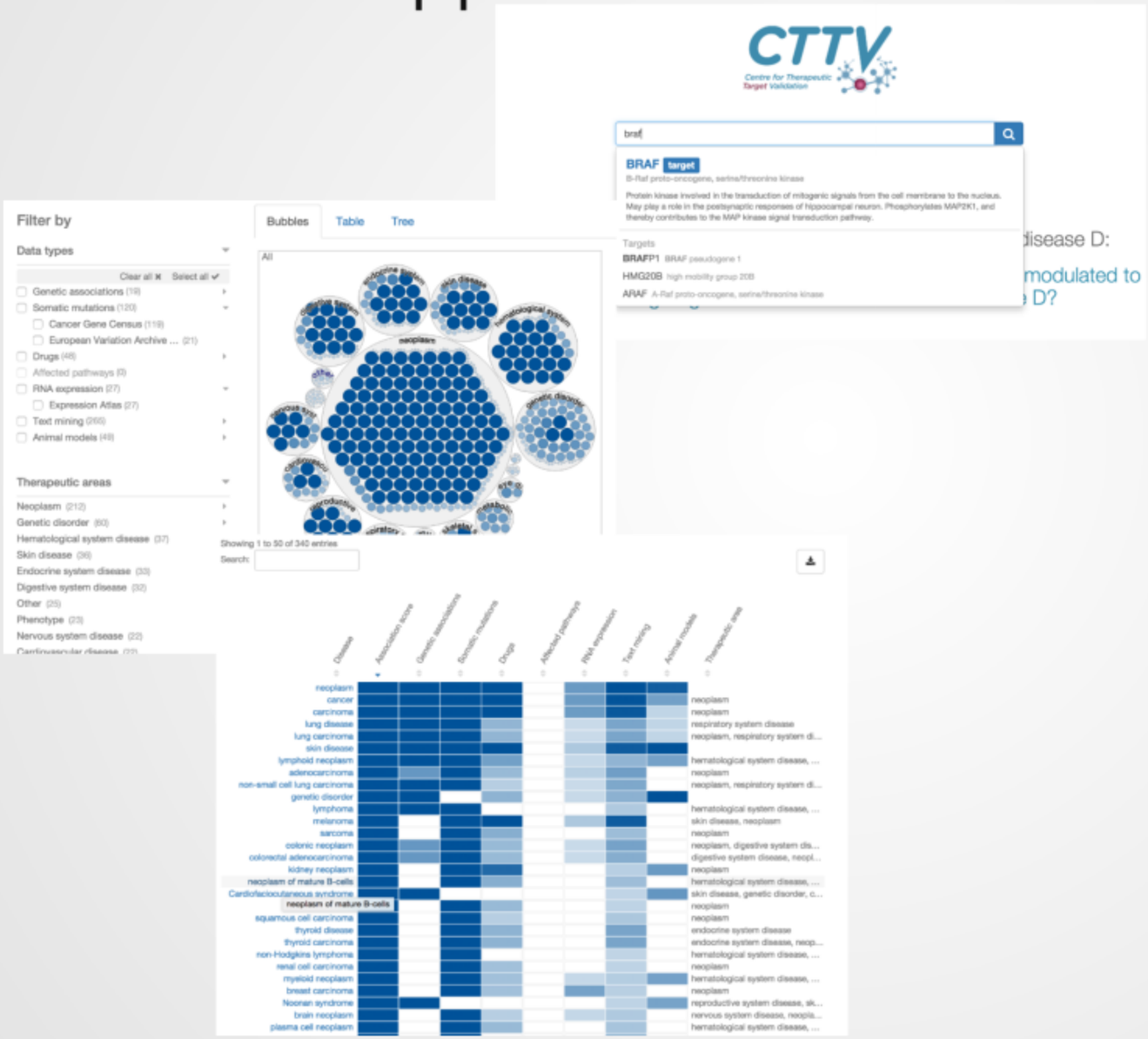
- Support autocompletion
- Disambiguate between different document types
  - interleukin
    - TARGET: interleukin-1, interleukin-2, ...
    - DISEASE: Immunodeficiency due to interleukin-1 receptor-associated kinase-4 deficiency
- Support for acronymns and many, many target synonyms
  - query: "SLE"
    - SLEep? (EFO term)
    - SLEED1? (Target)
    - systemic lupus erythematosus (what they were actually looking for)
- Data driven search score
  - Increase with the number of associations linked
  - Decrease if down in the ontology tree
- Track user behaviour?

REST API powers a search driven web app with rich visualisation

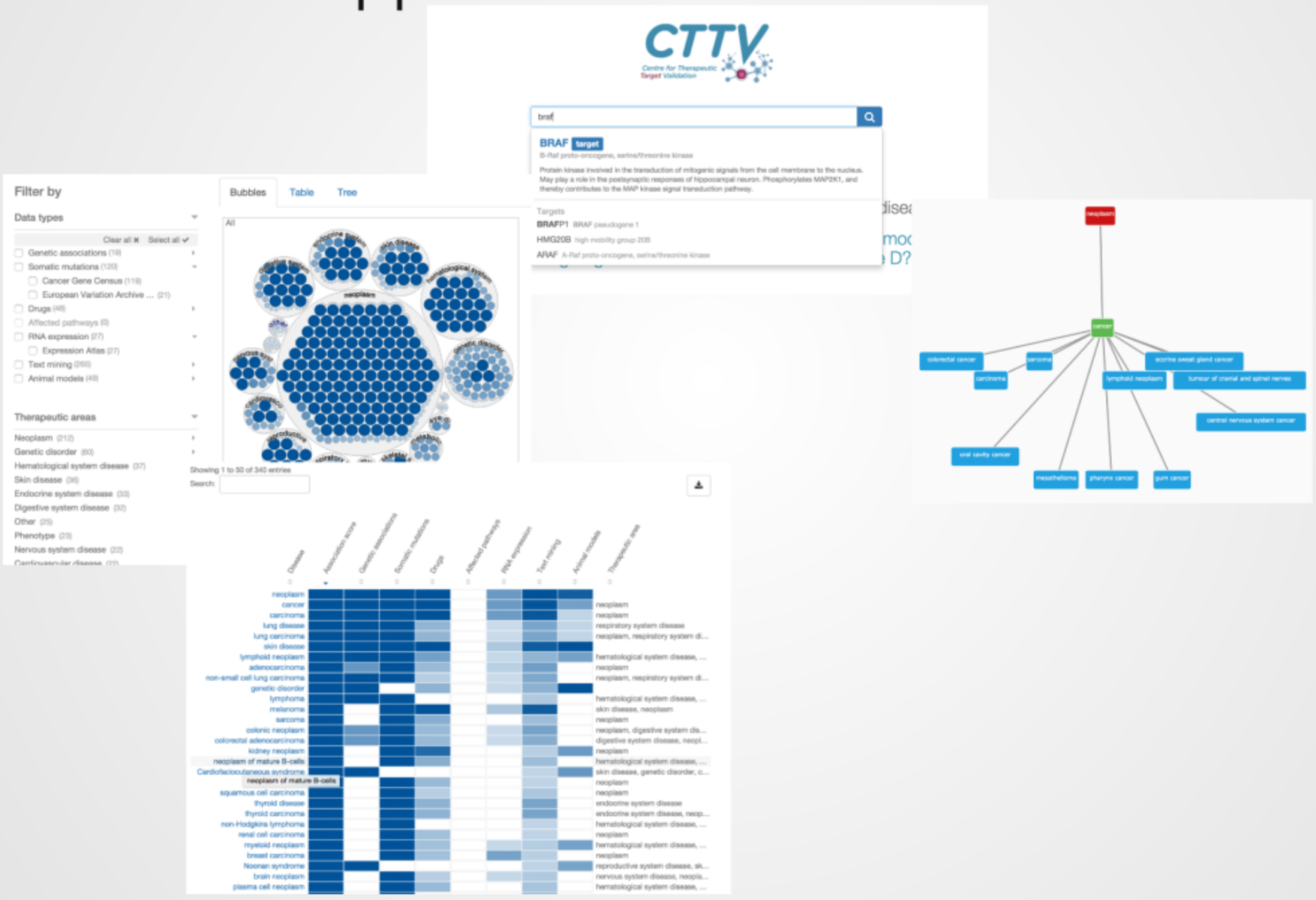
REST API powers a search driven web app with rich visualisation

REST API powers a search driven web app with rich visualisation

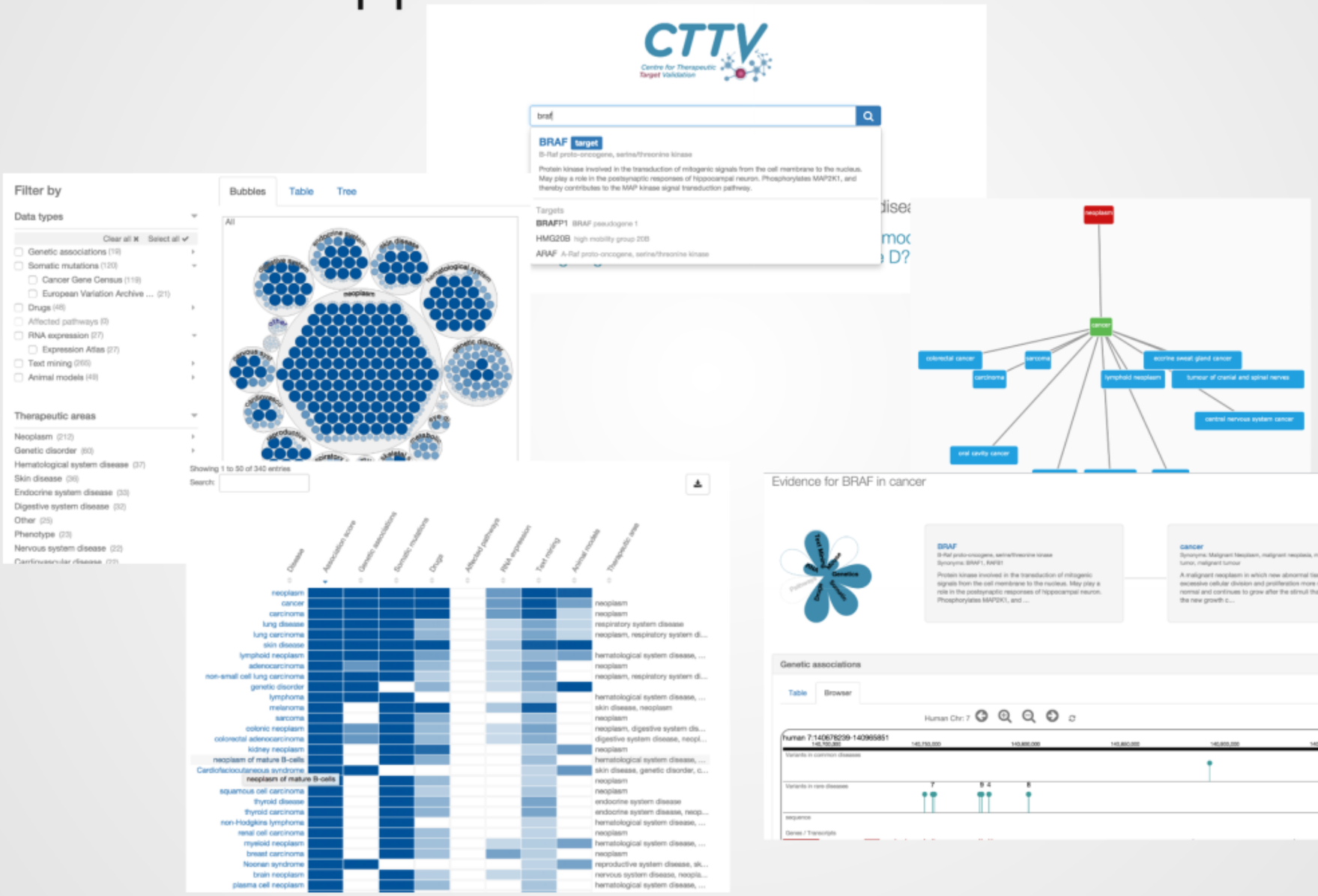
REST API powers a search driven web app with rich visualisation



REST API powers a search driven web app with rich visualisation



REST API powers a search driven web app with rich visualisation



Architecture

ES Cluster

- Running in the EBI Embassy cloud (VMWare)
- CoreOS + Docker
- 3 Nodes
  - Production: 4 CPU / 12 Gb RAM / 40 Gb SSD
  - Staging: 4 CPU / 8 Gb RAM / 100 Gb spinning disk

Architecture evolution

- Migrate from 1.7.2 to 2.1
- Drop production/staging paradigm and move to one big cluster
  - hot and cold nodes
  - tagged version of the data
  - support multiple version of data / api / webapp within the same infrastructure

Summary

Elasticsearch at CTTV

Elasticsearch at CTTV

Flexible, can easily handle many types of data

Main document datastore

Backend for computational pipeline

Powering REST API and WEB APP

Potential to run on demand map reduce computation


IT WORKS!

Acknowledgements

Core platform development team

Andrea Pierleoni

Gautier Koscielny

Ian Dunham

Jessica Vamathevan

Luca Fumis

Mick Maguire

Miguel Pignatelli

Niki Karamanis

Samiul Hasan

?