# cMapper Documentation

Muhammad Shoaib, Adnan Ahmad Ansari , Sung-min Ahn

Muhammad Shoaib
2016-06-16

# Documentation

In this section we present overall architecture of our framework which consists of four major components (1) Data Extraction (2) Data Cleansing or Preprocessing, (3) Relationship construction (4) Web UI.

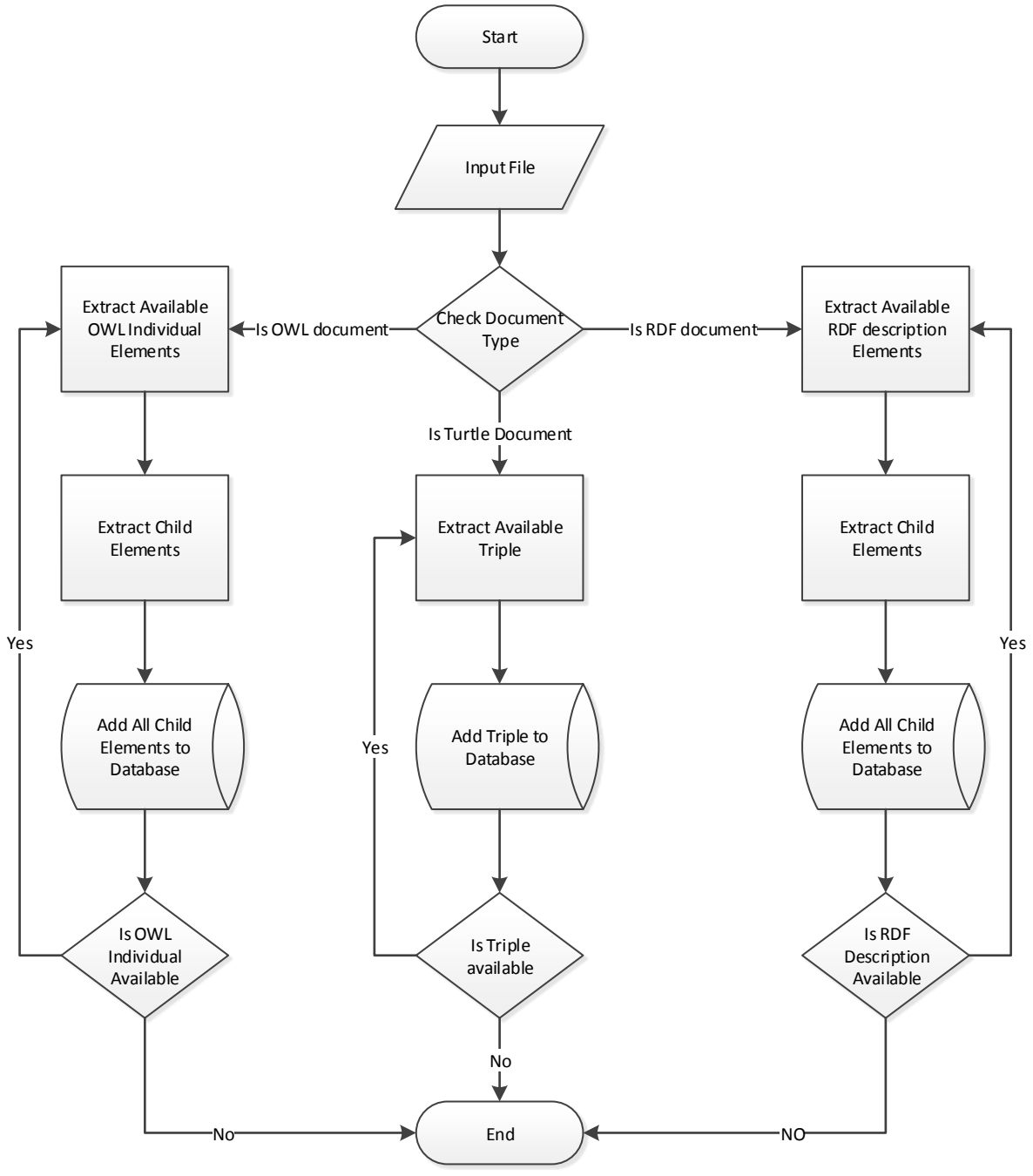



*Figure 1a: Flow Chart Representing Data Extraction Process*

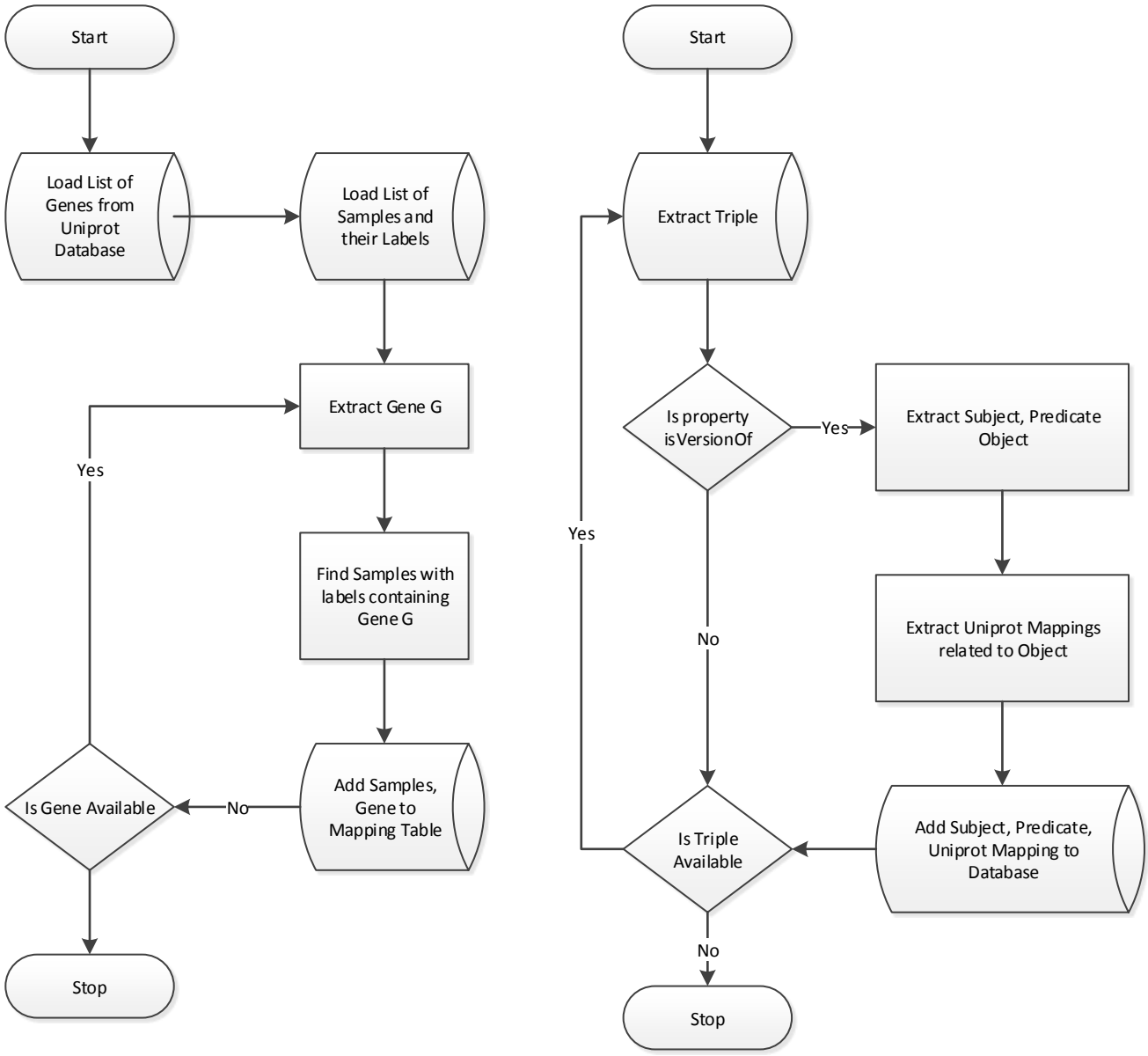



*Figure 2b: Flow Chart Representing Data Extraction Process*

## Data Extraction

Figure 1 shows the process of data extraction from different type of RDF documents. The process of data extraction includes (1) extracting data from EBI datasets, and (2) loading it into MySQL database. The major challenge that we faced in this module was diversity of data format. Although EBI claims that it has published data using Semantic Web Technologies i.e. (RDF) whoever they have not standardized their dumps so that they can be used into other applications. RDF dump of UniProt is about 136GB and it is in traditional Graph format so it was really difficult to load that RDF data into Database nor process that RDF file because of the hardware limitations. (Notice that our current work is not based cloud technology). On the other hand data about BioSamples is in the N-Triple format. Figure 3, Figure 4,

Figure 5 and Figure 6 presents examples of the data formats in which UniProt, atlas expression, bio samples and bio models data was available. From these figures it can be viewed that data was not in homogeneous format. Similarly the Graphical structures of REACTOME, Atlas and ChEMBL datasets are also not identical so that they can be parsed using simple RDF parser. We wrote six different parsers in Java to parse all six datasets and load them into MySQL database. Since our purpose was to link these six datasets and construct a connectivity graph therefore we only extracted the basic information from the RDF dumps that was sufficient for our tasks. Like we did not extracted biological processes and functions from the UniProt since this information can very easily be obtain by linking the platform with UniProt Database. Similarly we extracted the basic information from Expression atlas in the form of organism parts and assays and excrement details. For ChEMBL we extracted information about Small Molecules and Proteins and Drugs. From REACTOME Dataset we used chemical reactions at all level. Whereas BioSamples and BioMoelds datasets were completely used as they only contain meta-data information. The reason to use subsets was the simplicity at the current stage.

```
ID LCE6A_HUMAN Reviewed; 80 AA.
AC A0A183;
DT 15-JAN-2008, integrated into UniProtKB/Swiss-Prot.
DT 14-NOV-2006, sequence version 1.
DT 24-JUN-2015, entry version 43.
DE RecName: Full=Late cornified envelope protein 6A;
GN Name=LCE6A; Synonyms=C1orf44;
OS Homo sapiens (Human).
OC Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
OC Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
OC Catarrhini; Hominidae; Homo.
OX NCBI_TaxID=9606;
```

```
ID CCNC_XENLA Reviewed; 283 AA.
AC Q4KLA0;
DT 15-JAN-2008, integrated into UniProtKB/Swiss-Prot.
DT 02-AUG-2005, sequence version 1.
DT 27-MAY-2015, entry version 50.
DE RecName: Full=Cyclin-C;
GN Name=ccnc;
OS Xenopus laevis (African clawed frog).
OC Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
OC Amphibia; Batrachia; Anura; Pipoidea; Pipidae; Xenopodinae; Xenopus;
OC Xenopus.
OX NCBI_TaxID=8355;
```

*Figure 3 An example of UniProt Text Representation of Protein*

```
<http://rdf.ebi.ac.uk/resource/biosamples/sample/SAMN01055052> rdf:type
 owl:NamedIndividual ;
 biosd-terms:Sample ,
 rdfs:label E. coli KTE136 Genomic DNA sample Escherichia coli KTE136_A ,
 dcterms:identifier SAMN01055052 ,
 dcterms:title E. coli KTE136 Genomic DNA sample Escherichia coli KTE136_A ,
 pav:derivedFrom <http://rdf.ebi.ac.uk/resource/biosamples/repository-web-record/EBI+Biosamples+Database:SAMN01055052> ;
     <http://rdf.ebi.ac.uk/resource/biosamples/repository-web-record/ebi.ena:SRS346121>,
sio:SIO_000332 <http://rdf.ebi.ac.uk/resource/biosamples/exp-prop-val/GNC-SAMN01055052#0316b3050106570d3e4fb8b9d77c3c3a> ;
    <http://rdf.ebi.ac.uk/resource/biosamples/exp-prop-val/GNC-SAMN01055052#157d2ad03f45a403ba6d8909bd1ef817> ,
biosd-terms:has-bio-characteristic
 <http://rdf.ebi.ac.uk/resource/biosamples/exp-prop-val/GNC-SAMN01055052#c115374bde17ef6c1d997baa2c33d301> ;
 <http://rdf.ebi.ac.uk/resource/biosamples/exp-prop-val/GNC-SAMN01055052#9914abe3168a883712ab6366ac1efae9> ;
 <http://rdf.ebi.ac.uk/resource/biosamples/exp-prop-val/GNC-SAMN01055052#b8e2cfc7f5777e8628008851bb0aafac> ;
 <http://rdf.ebi.ac.uk/resource/biosamples/exp-prop-val/GNC-SAMN01055052#bee811c79426f216d654f91b5830f9ed> ;
 <http://rdf.ebi.ac.uk/resource/biosamples/exp-prop-val/GEN-SRA053203#9914abe3168a883712ab6366ac1efae9> ;
 <http://rdf.ebi.ac.uk/resource/biosamples/exp-prop-val/GEN-SRA053203#b8e2cfc7f5777e8628008851bb0aafac> .
```

*Figure 4 An example of BioSamples XML objects*

```
<owl:NamedIndividual rdf:about="http://purl.uniprot.org/uniprot/A0A183">
     <rdf:type rdf:resource="http://rdf.ebi.ac.uk/terms/atlas/UniprotDatabaseReference"/>
     <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A0A183</rdfs:label>
     <terms:identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A0A183</terms:identifier>
  </owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://rdf.ebi.ac.uk/resource/atlas/A-AFFY-11:64786_at">
     <atlas:dbXref rdf:resource="http://identifiers.org/ensembl/ENSG00000235942"/>
     <atlas:dbXref rdf:resource="http://identifiers.org/ncbigene/448835"/>
     <atlas:dbXref rdf:resource="http://purl.uniprot.org/uniprot/A0A183"/>
  </owl:NamedIndividual>
  <owl:NamedIndividual rdf:about="http://rdf.ebi.ac.uk/resource/atlas/A-MEXP-2246:2359420">
     <atlas:dbXref rdf:resource="http://identifiers.org/ensembl/ENSG00000235942"/>
     <atlas:dbXref rdf:resource="http://identifiers.org/ncbigene/448835"/>
     <atlas:dbXref rdf:resource="http://purl.uniprot.org/uniprot/A0A183"/>
  </owl:NamedIndividual>
```

*Figure 5 An example of Atlas XML objects*

```
  <rdf:Description rdf:about="http://purl.uniprot.org/uniprot/Q4KLA0">
    <owl:sameAs rdf:resource="http://identifiers.org/uniprot/Q4KLA0"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://identifiers.org/biomodels.db/BIOMD0000000003#_230475">
    <sbmlRdf:inCompartment rdf:resource="http://identifiers.org/biomodels.db/BIOMD0000000003#_230461"/>
    <sbmlRdf:name>Cyclin</sbmlRdf:name>
    <sbmlRdf:initialConcentration rdf:datatype=
     "http://www.w3.org/2001/XMLSchema#double">0.01</sbmlRdf:initialConcentration>
    <bqbiol:isVersionOf rdf:resource="http://identifiers.org/interpro/IPR006670"/>
    <sbmlRdf:constant rdf:datatype=
     "http://www.w3.org/2001/XMLSchema#boolean">false</sbmlRdf:constant>
    <sbmlRdf:initialAmount rdf:datatype=
     "http://www.w3.org/2001/XMLSchema#double">NaN</sbmlRdf:initialAmount>
    <sbmlRdf:boundaryCondition rdf:datatype=
     "http://www.w3.org/2001/XMLSchema#boolean">false</sbmlRdf:boundaryCondition>
    <bqbiol:isVersionOf rdf:resource="http://purl.uniprot.org/uniprot/Q4KLA0"/>
    <rdf:type rdf:resource="http://identifiers.org/biomodels.vocabulary#Species"/>
    <bqbiol:is rdf:resource="http://identifiers.org/biomodels.sbo/SBO:0000252"/>
    <sbmlRdf:hasOnlySubstanceUnits rdf:datatype=
     "http://www.w3.org/2001/XMLSchema#boolean">false</sbmlRdf:hasOnlySubstanceUnits>
    <sbmlRdf:substanceUnits rdf:datatype=
     "http://www.w3.org/2001/XMLSchema#string">substance</sbmlRdf:substanceUnits>
    </rdf:Description>
```

*Figure 6 An example of BioModel XML objects*

## Data Normalization

In most of the databases / triple store direct connections exists with UniProt however this is not the case for each and every dataset. Consider the example of BioModeles database where we have few UniProt identifications whereas links with other databases like Ensamble etc. exits. Since our framework is constructed by considering UniProt as the baseline therefore we had to convert those mapping into UniProt mappings because we took UniProt and Gene Names as the stranded for data normalization process. This process was undertaken by Data Normalization module of the framework. Luckily UniProt provides the mapping to proteins entities with other databases that was used for normalization of BioModeles. In the previous section we have explained that how we found the bridged properties between databases. In the process of data normalization we first extracted those bridged properties that bridge BioModeles with UniProt mainly. In the next step we check the diversity in the ranges of those properties which were identified as bridged. For each of object value of those properties we mapped them with available UniProt Values. After this exercise all non-UniProt identifier was replaced with UniProt identifiers. The same exercise was done with the BioSample Database the difference between normalization of BioModeles database and BioSample database was that BioSample was normalized based on Genes Names instead of UniProt Identifiers. There was no mapping available for

Gene Names therefore we have to map each sample-gene relationship manually. The process of mapping gene-sample was computationally expensive but was a onetime effort as for each gene we have to search for relevant samples and add them to mapping table.

## Connecting Object

This component can be said as the heart of the framework since it provides the core functionality. Once we extracted the data, standardized the URIs in the per-processing module, this module will construct the relationships amount different objects. The major challenge was to identify the common data points present in all six biological databases.

In the first step we took gene-protein network from Uniprot. Our tool allowed user to view that a particular gene is present in which organisms. Once obtained the gene-protein network we linked it with REACTOM database in the next step using Uniprot IDs that were present common in both datasets. Once connected with the REACOM one can find common reactions among two or more than two (up to 5) genes. Thirdly we connected the gene-protein-reactome network with expression atlas network. We found that expression atlas has relationships with both genes and proteins as well. Proteins relationships re directly mapped using RDF triples with dbXRef property whereas the genes are present in the labels so there were no triple presents in dataset linking expressions with the genes. With the gene-proteins relationship we can view in how many species a particular gene contributes in making the proteins whereas by analyzing expression atlas we can view that for which particular organism of a specie the gene has been regulated. By giving more than one gene we can also view whether there exists any organism in which given set of genes have been regulated together or whether the given set of genes have been experimented together or separately. After integration of atlas expression, we connected ChEMBL in our network. ChEMBL has direct connections with Uniprot dataset in the form of compound sequence. The most difficult part of the object connection adding bio-sample and bio-models in our network where no direct relationships with genes are present. We developed an algorithm for automatically link discovery from these databases with UniProt and Atlas Expression databases. The algorithm checks the triples having links from UniProt and Atlas Databases, mark the properties as bridge properties and add those triples to mapping table.

### Relation Discovery

Each biological database contains cross links with other databases that can be used to integrate them to explore new insights which are originally hidden and cannot been explored without connecting them with each other.

The core of our framework is extracting those relationships among these six databases that allow us their integration. in other words we have to find those properties in these triple stores which are present in more than one datasets. Since the name of the properties are not standardized therefore it was not possible to find the common properties based on their names. The other way can be using the domains and ranges of the properties. We have named these properties \emph{bridged properties} as they work as a bridge between two datasets. \emph{Bridged Properties} are those properties whose domain is in one dataset and range is in other dataset. Our relation discovery mechanism consists of three basic steps (1) Identifying the bridge properties, (2) Data Normalization (3) Network Construction and (4) Link Discovery.

## Identifying the bridge properties

As explained Bridge properties are those which has subject in one dataset and object in other dataset.

Let $T_1 = \{S_1, P_1, O_1\}$ and $T_2 = \{S_2, P_2, O_3\}$ be two different triple stores, a property $p_{\{11\}} \in P_1$ is called bridged if and if only if $Domain(p_{\{11\}}) \subset S_1 \wedge Range(p_{\{11\}}) \subset O_2$ or $Domain(p_{\{11\}}) \subset S_2 \wedge Range(p_{\{11\}}) \subset O_1$ or vice versa.

To identify the bridge properties we first loaded each triple store into MySQL database as explained in {Data Extraction} section. We then extracted all properties using {rdf:type} as for each property {?p} there exists a triple {?p rdf:type owl:Property} from each triple store and insert them into global properties table along with triple store identifier. This task can be accomplished by comparing the set of domains and ranges of each property with each other properties however this operation is computationally expensive and not possible. Consider two similar properties $p_1$ with domain $D_1$ and range $R_1$ and $p_2$ with domain $D_2$ and range $R_2$ and we have to find whether $p_1$ is similar to $p_2$ and vice versa. The time complexity will be $min(|D_1|, |D_2|) \times minn(|R_1|, |R_2|)$. Therefore instead of finding all bridge properties we first manually selected potential bridged properties who can play key role in interlinking two databases. For example for integrating atlas database with UniProt we searched for those properties that has link with either UniProt ID or Gene Name. Similar kind of exercise was done for ChEMBL and REACTOME databases. For Biosamples we extracted properties that contains links with Gene Name and Expression Atlas.

Once we identified the bridge properties, standardized them, the next step was the standardization the names of their domains and ranges since they varies from database to database.

## Network Construction

Since these all triple stores were not constructed by single community therefore there dumps are available with an identical Ontology. To add these triple stores in our biomedical knowledgeable we have to normalize their IRIs. In the first step we remove all the prefixes from the properties by replacing with their real IRIs. The next step was for Atlas, Bio Samples and Bio Models in which we add mapping triples for genes and proteins as no triple describing the relationships of samples, models and expressions with genes were present in the original triple stores. To do so, we identified the potential properties whose object may have the names of genes and proteins. As these properties does not lead to direct required objects, i.e. samples or models so for each identified property we manually traced the hierarchy which leads to either sample model or experiment. We then constructed the separate queries for each identified properties to construct the links between the objects of identified properties (i.e. genes or proteins) and expressions, models and samples. By the end of this exerciser we had information that which expression, sample and model is linked with which genes and proteins.

## Link Discovery

This is final step that is performed on the real time by our web application. Given more than two genes or small molecules this module runs a greedy algorithm to check whether there exists any common links among the given genes and small molecules or not. The algorithm works as following; for each gene it extracts its all present links in the network and stored in local array. One array is created for one gene. Once links for all genes are extracted and stores our greedy algorithm finds the common nodes in all arrays which are then present to the respective user on the web browser in the form of graph.

## Web Portal

Web portal was the last step of framework development which provides users an interface to search for any particular gene or small molecule to get its links in other databases. As no such prototype is available already that can be borrowed so we developed in house java based portal to provide this functionality to the users. When a user input a gene or small molecule and select particular databases, as the output he gets a graph with the data points as nodes and links among those nodes as edges. Nodes belonging to different databases are colored differently to ensure readability. We used Cytoscape (a well-known software for graph construction)'s web API to constitute the graphs.

Figure 7 and Figure 8 show the procedural pipelines that web portal performs after receiving a gene or small molecule as query. For a small molecule in the first step list of those proteins and genes which are regulated by given drug are extracted. In the second step list of proteins are given as input to expression atlas, CheMBL, REACTOM and BioModels and genes to BioSamples to extract all possible entities associated with gene(s) and their proteins. After extraction of entities from all databases they are fused to create one comprehensive graph. The resulted graph is then transferred into GraphML and return to user agent (Web Browser) which store the GraphML or create a visualized graph from it according the users' requirements.

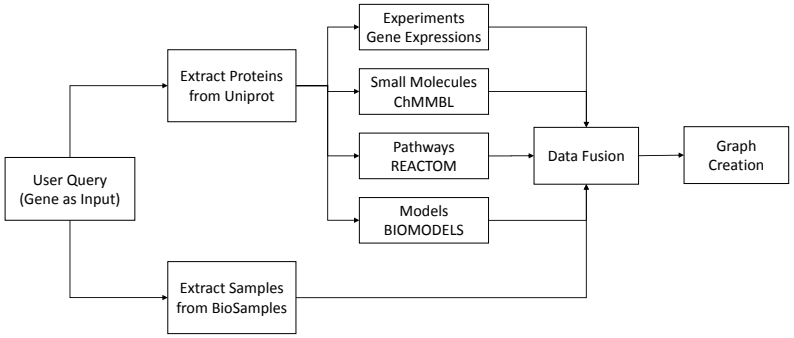



*Figure 7 Procedural pipeline of graph creation for a User Query (Gene)*

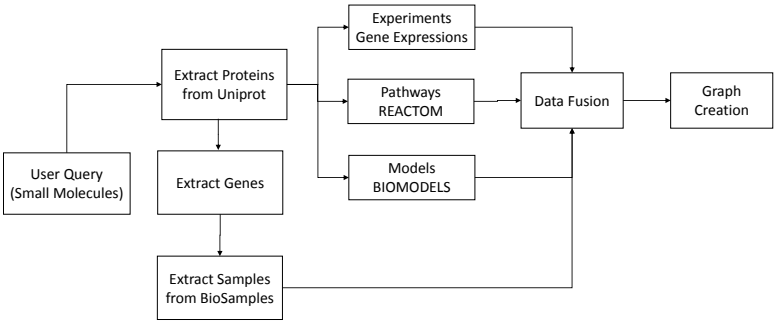



*Figure 8 Procedural pipeline of graph creation for a User Query (Drug)*

# cMapper User Queries

In the following supplementary tables we have presented details about queries that we inputted to the cMapper for generation of figure 1 to 3. Readers can use the information given in the following tables to regenerate the similar graphs as given in the paper.

Table S1: User Input query for Figure 1

| **Input Type** | | **Input** | | |
|---|---|---|---|---|
| Gene | | CTNNB1; STAT3; FGFR4 | | |
| **Databases Included** | | Associated Genes, UniProt, Expression Atlas, REACTOME, ChEMBL, BioModels, BioSamples | | |
| **Databases Excluded** | | None | | |
| **Filters** | | | | |
| **Organism Filter** | **Organ Filter** | **Pathway Filter** | **Graph Type** | **Output Type** |
| Homo Sapiens | All Organs | All Pathways | All Connections | View Graph |

Table S2: User Input query for Figure 2

| **Input Type** | | **Input** | | |
|---|---|---|---|---|
| Gene | | CTNNB1; STAT3; FGFR4 | | |
| **Databases Included** | | UniProt, Expression Atlas, REACTOME, ChEMBL, BioModels, BioSamples | | |
| **Databases Excluded** | | Associated Genes | | |
| **Filters** | | | | |
| **Organism Filter** | **Organ Filter** | **Pathway Filter** | **Graph Type** | **Output Type** |
| All Organisms | All Organs | All Pathways | Shared Connections | View Graph |

Table S3: User Input query for Figure 3

| **Input Type** | | **Input** | | |
|---|---|---|---|---|
| Gene | | CTNNB1; STAT3; FGFR4 | | |
| **Databases Included** | | Associated Genes | | |
| **Databases Excluded** | | UniProt, Expression Atlas, REACTOME, ChEMBL, BioModels, BioSamples | | |
| **Filters** | | | | |
| **Organism Filter** | **Organ Filter** | **Pathway Filter** | **Graph Type** | **Output Type** |
| All Organisms | All Organs | All Pathways | Shared Connections | View Graph |

# cMapper User Guide

Muhammad Shoaib, Adnan Ahmad Ansari, Sung-min Ahn

Department of Biomedical Engineering

College of Medicine, University of Ulsan, (Asan Medical Center, Seoul)

The cMapper home page is very similar to any other search engine home page that provides users a state forward view to input genes or small molecule and select the filters for the graph generation. Figure S1 and S1A show the cMapper homepage which consists of filters, gene or small molecule selector, database selector and input text box.

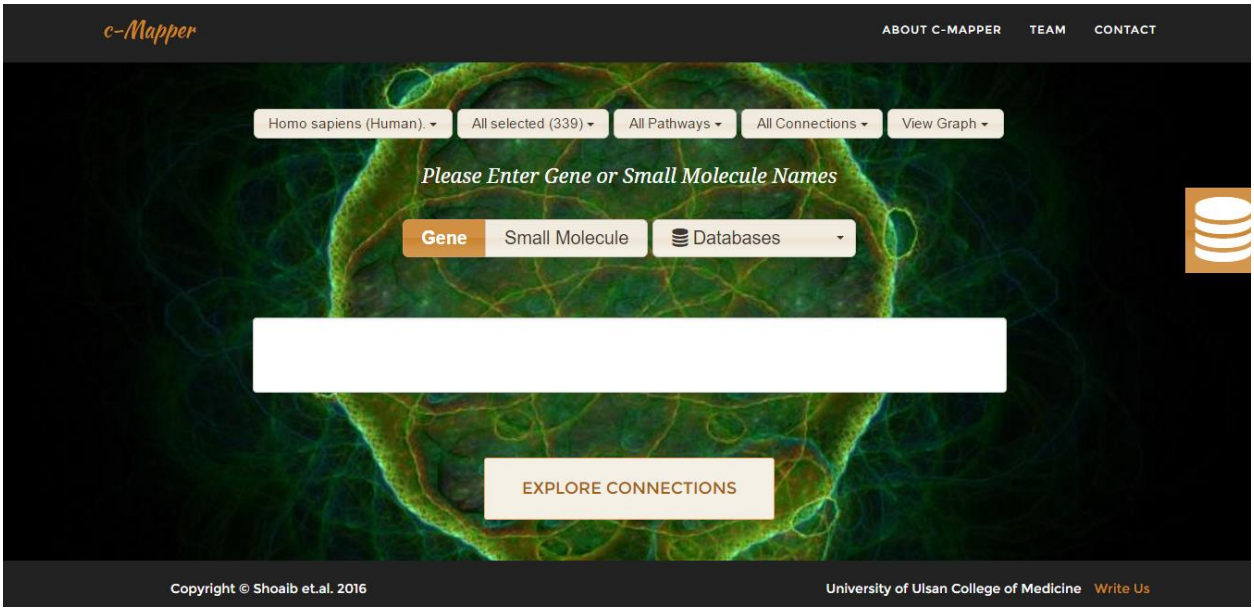



Figure S1: cMapper Homepage.

When user enter list of gene of small molecule the output of cMapper is graph figure S2 shows the cMapper graph displaying window. On the top the list of genes or small molecules that had contributed towards creation of graph are sown. In the middle of the screen the graph and bottom is general footer.

User can edit the list of genes by clicking on the “edit” button and download the graph by clicking on the “download” button. Upon selecting the edit option screen similar to figure S3 appears which allow users to alter the list of genes or small molecule as well alters the filtering options as well. Furthermore figure S4, S5 and S6 shows the screen short of various filtering options

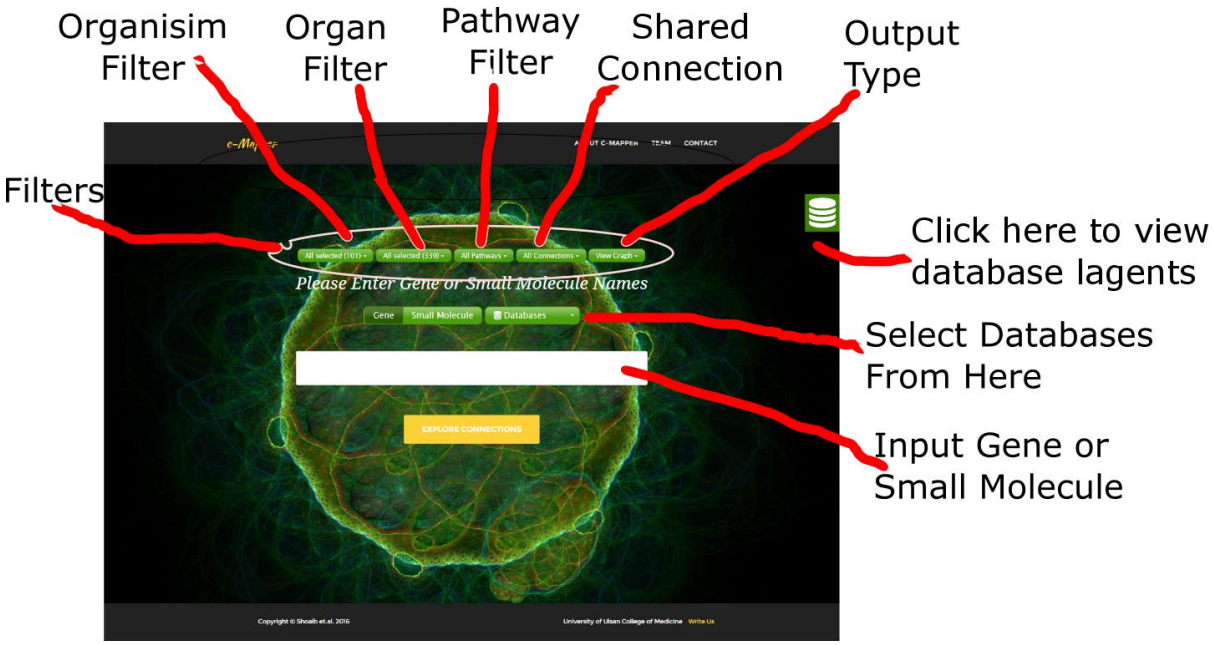



Figure S1A: Components of cMapper Tool

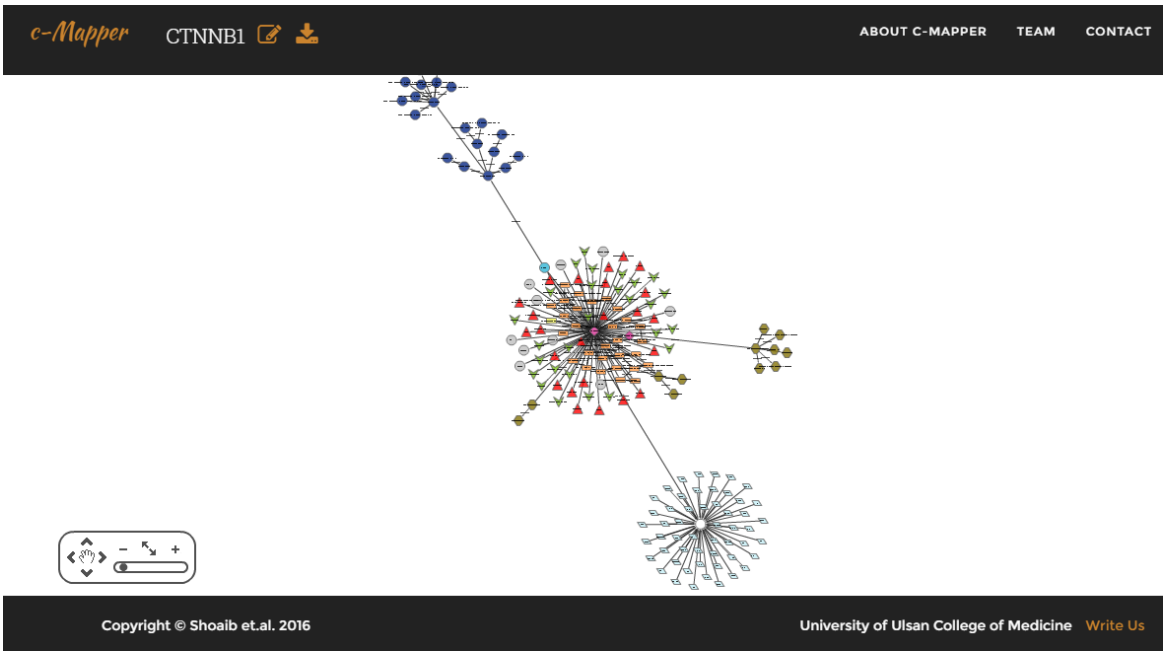



Figure S2 cMapper output windows which includes list of gene names – “CTNNB1” in this case –, edit button, and download button. The “edit” button can be used to edit list of gene names or update filters whereas “download” button can be used to download the graph as PNG file.

In the following we have shown the details about filters andtheir functions along with screen short for each filter.

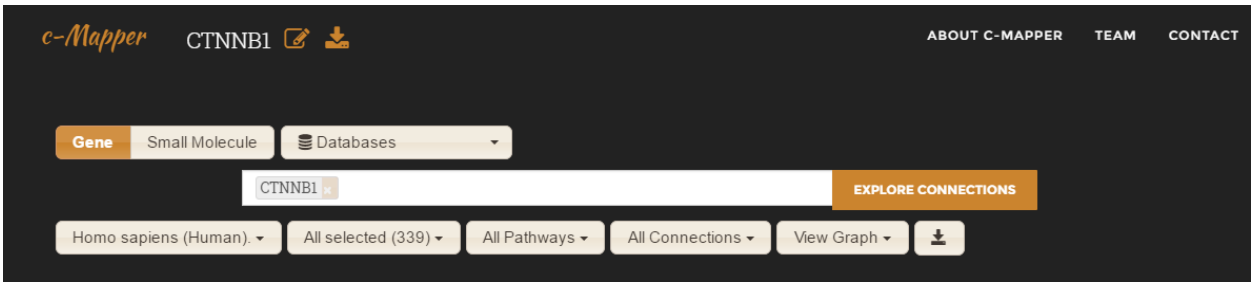



Figure S3: cMapper editing window that appears after clicking the edit button.

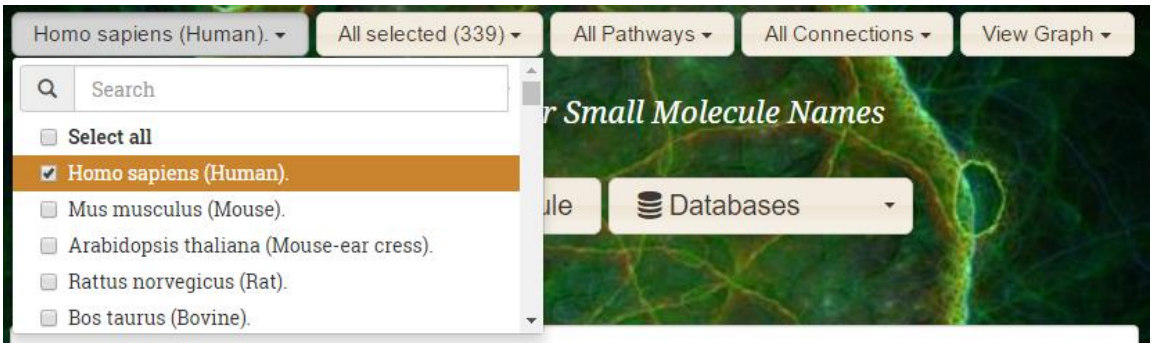



Figure S4: cMapper organismic filter. By default Homo sapiens (Human) organism is selected users can edit and select multiple organisms of their choices. The list is sorted based on number of proteins against organisms in the UniProt KB.

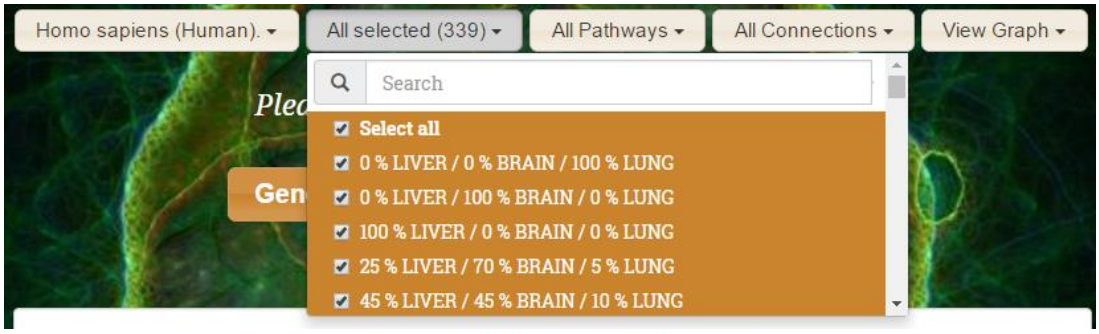



Figure S5: cMapper Organ selector that users can use to specify the list of organs like “brain”, “liver” etc. This list is sorted based on Alpha-numeric letters. By default all organs are selected users can unselect organs of their choice as well.

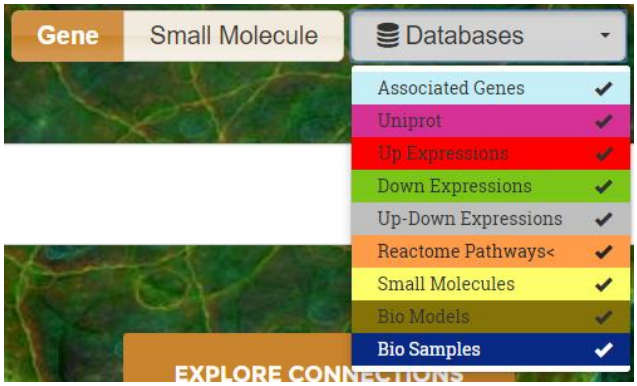
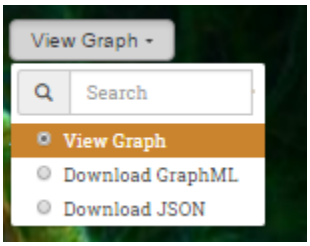



Figure S6: cMapper database selector that allows the users to select the database of their choice. When a user deselected a database the output graph will not contain the database entities from the deselected database.


All and shared connections filter


Metabolic and Signaling pathway filter




Output Option

Similarly other filters includes “Metabolic” and “Signaling” pathways filter, and “Output” option that allows users to select their output type. These output types includes “view graph” – graph will be displayed on the user’s screen –, “Download GraphML” – graph will be downloaded on the user’s device in the form of GraphML – and “Download JSON” – graph will be downloaded on the user’s machine in the form of JSON Array.