Function:Troilkatt/Additional Data Structures

From FunctionWiki

This document contains descriptions of additional Troilkatt data sources and data formats that have not been yet implemented:

Contents

Assembly Archive

We have also plans to download sequence data from the NCBI Assembly Archive (or can we also use the Trace Archive or Short Read Archive?). Sequencing data differs from Microarray data in that it does not contain expression values. However, we can obtain expression values by counting the number of short read sequences that maps to a particular gene under a given experiment condition (and by taking into account their quality scores). This allows creating a PCL file.

Additional repositories

Additional databases that could be listed are (from Hefalamp):

- BioGrid. About 100MB of XML, tabular text files, and Osprey data files.
- BOND/BIND (Biomolecular Object/Interaction network databank). No way to download entire database?
- Database of Interacting Proteins (DIP): ~100 MB of XML, FASTA, MIF, and tabular text files. About 900(?) datasets, but in a single file.
- Gene Set Enrichment Analysis (GSEA): a few MB of GMT files.
- EBI IntAct databse. ~500MB of ?
- Molecular interaction database(MINT): A few tens of either XML or tabular text files.
- Protein-protein interaction database: less than 1MB.
- PFAM-B database: ~50GB.
- ProSite database: 1MB text file.

Currently the only meta data source is SGD, but it has only information for yeast. In addition there are manually downloaded files. Possible sources for meta data are:

- Entrez Gene gene_info files seems to provide the information about “current” genes, while gene_history has gene names no longer in use.
- UniGene for list of gene names
- BioMart
- Pathway interaction database (PID)
- Reactome – pathways
- Human protein reference database

- KEGG
- GO

DAT

For query and exploration, the metric of interest is usually which genes are correlated with which other genes. These gene-to-gene correlations for a dataset are calculated using a distance metric, f(z) in Spell paper (http://bioinformatics.oxfordjournals.org/cgi/content/abstract/23/20/2692) , and stored in a **DAT** matrix file:

DAT (gene distance matrix) file format.

	gene_1	gene_2	gene_3	...	gene_G
gene_1		distance-1-to-2	distance-1-to-3	...	distance-1-to-G
gene_2	distance-2-to-1		distance-2-to-3	...	distance-2-to-G
gene_3	distance-3-to-1	distance-3-to-2	-	...	distance-3-to-G
...					...
gene_G	distance-G-to-1	distance-G-to-2	distance-G-to-3	...	-

HBase DAT

Each distance matrix can be saved as a HDFS file. However, the common access pattern for Spell search is to only read a few rows query genes related in each matrix. HDFS is not designed for such access patterns, but Hbase is. We therefore store all distance matrices in a HBase table with the following format:

Distance matrices big table.

Row-key	Distances:
dataset_1-gene_id1	{t1 = [to-gene_2, to-gene_3, to-gene_4,...,to-gene_G], t2 = [to-gene_2, to-gene_3, to-gene_t,...,to-gene_G]...}
dataset_1-gene_id2	{t1 = [to-gene_1, to-gene_3, to-gene_4,...,to-gene_G],t2 = [to-gene_1, to-gene_3, to-gene_4,...,to-gene_G]...}
..	
dataset_D-gene_idG	{t1 = [to-gene_1, to-gene_2, to-gene_3,...,to-gene_G-1], t2 = [to-gene_1, to-gene_2, to-gene_3,...,to-gene_G-1]...}

All gene-to-gene distance arrays are timestamped by HBase allowing us to store multiple versions (if needed). Addition, update, and deletion will be done by HBase row manipulations. However, since the per-dataset gene-to-gene distances only change if the raw data is changed there is rarely a need to modify the values, so there are typically few versions.

The row identifier is the concatenation of the dataset name and gene id. This ensures that rows from the same dataset are stored together. There is only one column family distances: which has a single column qulifier (empty name) where all the gene-to-gene distances are saved.

The size of a row will be about 100KB for a genome with 25,000 genes.

Estimated human compendium file sizes

File type	SOFT	PCL	Intermediate	DAT	Lookup table
Dataset size	?	14GB	85GB (6xPCL)	3.5TB	3.5TB

In the above table the data size of the target human compendium is estimated. The intermediate file and size is estimated by assuming that each file has 100 samples and that each expression values is encoded using 4 bytes. In practice these files are larger since they are stored in text format.

The lookup table can be created by running the mrDat2Ddb program <which is not yet implemented>, that will create or

update lookup table. Alternatively a Java interface can be used:

```
LookupTable:
  addDatasets(geneX, geneY, metricName, datasetNames[],
   datasetDistances[], userID, timestamp)
  update()
  delete()
  datasetNames[]: readDatasetNames(timestamp, userID)
  distances[]: readDatasets(geneX, geneY, metricName, userID,
   timestamp)
```

Lookup Table

A compendium is stored in a **lookup table**:

DDB lookup table format.

gene-pair	distances
(gene_1, gene_2)	[distance_d1, distance_d2, ..., distance_e]
(gene_1, gene_3)	[distance_d1, distance_d2, ..., distance_e]
...	...
(gene_g-1, gene_g)	[distance_d1, distance_d2, ..., distance_e]

This lookup table is to reorganize .DAT data and store only one gene pair in row. It is basically a distributed version of the Sleipnir .DB files. It is also supports addition, updates and deletion of datasets.

Open issues:

- What should each cell contain? A single distance value, or an array with all gene-to-gene distances from all datasets? With the former adding new datasets is easier, but the later will probably have much better performance for the performance critical search stage. Need to measure overhead of reading a different types of rows from Hbase.
- How many gene-to-gene distances should there be per row? Note that current implementation of Hbase reads (including decompress) an entire row at a time even if only a single column is needed. This requires finding the right tradeoff between multiple Hbase lookups and the overhead of reading larger rows (where possibly only a subset of the values are needed).
  - Assuming there are 1500 datasets, with a 4-byte float per distance value the array size for a gene-to-gene pair will be less than 6KB. If there are 25.000 genes, for each gene the distance arrays will be 143MB. There are about 625.000.000 rows.
- How sparse are the matrices? Perhaps a more efficient data structure is to not save any zero elements. Also, are the matrices sparser for Spell than other algorithms?
- Design with support for addition/deletion of datasets, or just rebuild everything? Depends on time to rebuild a dataset

Proposed design:

```
Lookup-table:
  'gene_id1-gene_id2-distance':
    public:bin1:
      timestamp: [d1, d2, d3,...,dk],
    public:bin2:
      timestamp: [dk+1, dk+2, dk+3,...,dk+k],
    ...
    private:uid1:
      timestamp: [pd1, pd2, pd3,...,pdl],
```

There is one row per gene-to-gene pair . The gene-names are used as row-key. There are two column families: public and private. The public distances are saved in arrays but are split among multiple bins each implemented as a column in Hbase, such that each cell has k distances. The private datasets are stored in the private column family under a column identified by the user ID. The following operations are supported:

- To *read* the distances calculated for a version of the compendium, a timestamp is provided and used to return the correct version of each column.
- The receiver must ensure that private data is removed. Alternatively this may be done by a library function in the provided API.
- To *add* a public dataset the array in the newest bin is appended, or if it is full a new bin is created. This reduces the amount of data that must be read and written to the table. TODO: find right balance.
- To *add* a private dataset a new column is created in the private column family.
- To *update* a dataset the bins of each row containing the datasets distance value is read, updated, and saved under a new timestamp.
- To *delete* a public dataset the dataset value is set to NaN in each row.
- To *delete* a private dataset the column is deleted.

A possible Spell optimization

One query-gene pre-computed data not that interesting. But can it be used to optimize the queries? Actually, for human genome, the intermediate DAT files will be 3.75TB. Since we want to reorganize them to hbase system, according to its 3 duplicate rule, we will need 11.25 TB. Each query result is about 220KB and can be compressed much for zero values. So, for 11.25TB we can store at least 51M query results, which is hopeful to store all the 1 and 2 human genes query results. It is better in design decision to allocate some space as search results hash buffer. Then the user can expect to get most query results immediately.