



Universiteit Leiden

Computer Science

A Metadata Validation Process Design for an
Automated High-Throughput Screening Workflow -
Case Study in Metadata of CytomicsDB

Name: Zhihan Xia

Date: 27/08/2014

1st supervisor: Fons J.Verbeek

2nd supervisor: Enrique Larios

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)

Leiden University

Niels Bohrweg 1

2333 CA Leiden

The Netherlands

A Metadata Validation Process Design for an Automated High-Throughput Screening Workflow - Case Study in Metadata of CytomicsDB

Zhihan Xia (1244205)
Computer Science,
LIACS
Leiden University
The Netherlands
xzh1@live.cn

Dr. Fons. J. Verbeek (Supervisor)
Section Imaging and Bioinformatics,
LIACS
Leiden University
The Netherlands
f.j.verbeek@liacs.leidenuniv.nl

Enrique Larios (Supervisor)
Section Imaging and Bioinformatics,
LIACS
Leiden University
The Netherlands
e.larios.vargas@liacs.leidenuniv.nl

Table of Contents

1. Introduction	3
2. Validation Strategies	4
2.1 Definitions	4
2.2 Strategies	5
2.2.1 “Trust Your Friends” and “Pass It On”	5
2.2.2 Levenshtein distance	7
2.2.3 Multi-objective Decision	9
3. Validation Subjects	9
3.1 Treatment/Compounds	10
3.2 siRNA	10
4. Validation Workflow	12
4.1 Treatment/Compounds Validation	12
4.2 siRNAs Validation	17
5. the Architecture	24
5.1 The presentation layer	24
5.2 The utility layer	27
5.3 The service layer	27
5.4 The persistence Layer	28
5.5 The repository	28
6. Evaluation of the results	29
6.1 Accuracy of locating contradictions	30
6.2 Accuracy of locating duplication	31
6.3 Accuracy of giving solutions	31
7. Future Works	32
7.1 Duplex detection for multi-attributes entities	32
7.2 Multiple external data sources	32
7.3 Sequence correction	33
7.4 Multi-objective decision	34
7.5 siRNA correction	34
Reference	35
Appendix	36
RNA sequence links	36
The class diagram of siRNA validation process	37

Abstract. High-Throughput Screening (HTS) techniques are commonly used to identify potential drug candidates by applying screening strategies on large-scale small molecules and genome-scale RNAi. HTS experiments are mechanical and repetitive by nature with large volume of data involved. Some HTS experiments management applications have been developed as automatic data management and analysis solutions to cope with repetitive steps and data volume. Computation and transferring of data are automated by these systems to reduce the risk of errors which usually caused by unnecessary repetition of researchers during these stages. However, little attention has been paid to the consistency, integrity and reliability of preset parameters used in experiments. However, if these metadata are not trustful, then no matter how accurate it could be during data processing, no correct conclusions can be retrieved. Thus, an effective progress to validate HTS experiments metadata is highly needed to solid the foundation for the experiments and then potential candidates can be expected.

This thesis is going to propose a process to validate metadata (siRNA entities and Treatment/Compound names are going to be taken as use cases) during the pre-design stage of HTS experiments. The process is going to be performed when Master Tables for these parameters change (insert, update or delete entries) on CytomicsDB [1], which is a web based HTS workflow management platform runs with a modern RDBMS (Relational Database Management System).

1 Introduction

It is simply not possible to analyze the large amount of potential drug candidates available in the fields of biology and chemistry today through manual labor. The process of lead screening needs to be automated and its throughput increased if good new drug leads are expected to be identified within reasonable time frames and at reasonable cost. Then, using robotics, data processing and control software, liquid handling devices, and sensitive detectors, the methods of HTS are introduced to help researchers quickly assay and screen millions of chemical or genetic targets at a time. Through this process one can rapidly identify active compounds, antibodies or genes which are starting points for drug design.

Usually HTS experiments follow the steps of (1) plate preparation, (2) reaction observation and (3) "screening". Parameters like chemical treatments, siRNAs, type of plates, type of microscopes are pre-stored and maintained as libraries and carefully catalogued. During the plate preparation stage, researchers design experiments by setting metadata for each microplate. Metadata specified for each experiment (such as a protein, cells, or an animal embryo, and the treatments conduct upon them) are selected from the metadata libraries. Then, according to the experimental design, each well of the

plate is automatically or manually filled with specified cell populations and designed treatments are induced into each population. After some incubation time has passed to allow the biological matter to absorb, bind to, or otherwise react with the compounds in the wells, measurements are taken across all the plate's wells. Using the preset parameters as parameters, Elementary measurements are automatically conducted on time-lapse images taken by microscopes. These images show changes or defects in embryonic development caused by the Treatment applied in each well. Based on the measurement result, researchers can do more assays and select liquid from wells that gave interesting measurement results. The selected liquid will be put into other plates for following screen experiments. By collecting further data on the narrowed set in the following experiments, researchers can continually confirm and refine observations.

CytomicsDB system integrates the whole HTS workflow. The system relies on a modern relational database system, MonetDB [2], to store metadata and experiments' results, while providing a web base GUI for end-users to supervise and interact with data. During each stage of the HTS workflow, CytomicsDB involves a validation process to normalize metadata and intermediate data. This thesis presents the validation process which uses external databases to check the consistency of each metadata entry. The validation process is conducted during maintaining (adding, updating or deleting) metadata in master tables in CytomicsDB.

Two kinds of metadata, Compound names and siRNAs, are taken as cases for the validation process as they are representative (the Compound name is single-attribute metadata while siRNA is a multi-attributes one) among all kinds metadata stored in CytomicsDB. The volume of metadata vocabularies of these two categories also determines that they mostly need the auto-validation process. The volume for each of these two metadata is over thousands of entries which is too big to check manually. The thesis is going to discuss this validation process from the following aspects in section 2 to 5: (1) validation strategies; (2) validation subjects; (3) Workflows of the validation process and (4) the architecture of the validation process. An evaluation of the effectiveness of the validation process is going to be discussed in section 6. Some future improvements are going to be discussed in the final section.

2 Validation Strategies

2.1 Definitions

The validation process can be abstracted as a model. In this model, each object which is objectively existed in the real world (e.g. a compound, or a siRNA) is defined as an *entity* E . For each *entity*, several attributes are assigned to it, like the name, the ID number and the publisher. The *attributes* that are used to describe one *entity* can be defined as a set: $A = \{a_1, a_2, \dots, a_n\}$, in which a_i ($1 \leq i \leq n$) means the i^{th} *attribute* of the *entity*.

In this model, multiple data sources are involved as well. They can be categorized as two

types. One set is from the lab in which researchers use CytomicsDB to manage their experiment data. In CytomicsDB, this set is uploaded by researchers and stored as master tables in the database. Another group of sources are from external databases. They are used to validate the metadata uploaded by researchers. All these *data sources* can be expressed as a collection $S = \{s_1, s_2, \dots, s_m\}$ in which s_i ($1 \leq i \leq m$) represents the i^{th} *data source* among the m *data sources*.

Adopted from [3], The data source s_i offers a *fact value* $f_{(s_i, a_j)}$ for the attribute a_j of an *entity* E . different data sources may have different fact values for a same attribute of the entity. For the *entity* E , if $\exists a_j \in [a_1, a_n], f_{(s_i, a_j)} \neq f_{(s_l, a_j)}, i \neq l$, then a confliction or inconsistency is found between *data source* s_i and s_l . In all fact values from all data sources, those who correspond to the attribute value in the real world are called the *true value*. So the validation process is in fact a progress for identifying *true values* among all conflictions between data sources.

2.2 Strategies

2.2.1 “Trust Your Friends” and “Pass It On” [4]

The relationships among the entity, its attributes and fact values from different data sources in CytomicsDB are sketched in Fig. 1. In CytomicsDB, the idea is to validate the metadata while researchers building the metadata master tables in the system. In Fig 1, it means that the data source S2 from the researcher needs validation. An assumption is considered that the *fact values* from reliable external databases can be treated as *true values*. Especially when those *fact values* are identical to the ones given by researchers, the possibility that those fact values are *true values* becomes quite high which can be assumed as 100%. This confliction avoidance strategy is referred as “*Trust Your Friends*” in [4]. The intuition behind this strategy is to trust some data sources which are most reliable, data-rich and independent to the data source which needs to be validated. What sources to trust is decided once and carried out for all data values.

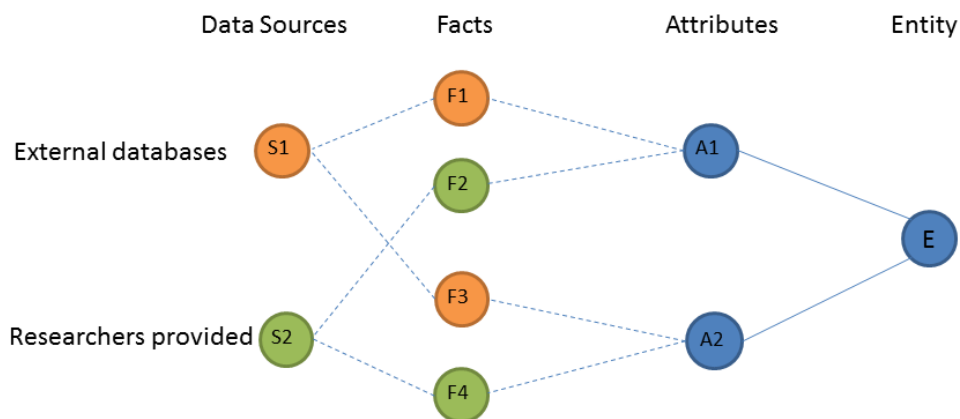


Fig. 1 the relationships between the entity, its attributes and fact values from data sources

One may have noticed that the fundamental assumption is a little bit arbitrary. Theoretically, no data sources can be 100% accurate in describing all entities in the real world. Since the researchers should be experts to the metadata they upload, the researchers' decisions are involved as a part of the confliction resolving strategy. The validation result from "Trust Your Friends" strategy is "Passed On (PASS IT ON [4])" to researchers (users, experts) to let them decide how to handle possible conflicts. For an entity, the validation result includes the status of its correctness and some possible solutions to handle conflicts when some conflictions are detected in some attributes. To the entity, the "Highest Quality [5]" entries (in all attributes) from external databases are given as recommended possible solutions to conflicts. The researchers have the final word on the confliction resolutions.

For entities with only one unidentifiable attribute, additional fact values of identities from external data sources should be used in the validation strategy. For entities with several attributes, some multi-objective decision strategies can be implemented during selecting the "Highest Quality" matchers.

The strategy can be mathematically expressed as following. [4] For a single-attribute entity, the conflict handling strategy received n entities from external data resources beside the entity itself. The strategy can be expressed as a function f_{ch} defined on a domain (the single attribute of the entity E) D and maps $n+1$ (the additional 1 input c represented the one fact provided by researchers) input values to one output value of the same or another domain S . Fact values (c_i) conflicted from the fact value (c) provided by researchers in the entity are resolved to a solution s :

$$f_{ch}: D^{n+1} \rightarrow S \quad (1)$$

$$f_{ch}(c_1, \dots, c_n, c) = s, s \in S, c_i \in D, \forall i = 1 \dots n$$

Similarly, an $n+1$ -ary multi-attributes conflict handling function is a function f_{ch} defined on m domains D_i and maps n input m -tuples to one output value s which is from the same or another domain S . The idea here is that conflicts are resolved in an attribute by using additional knowledge from other attributes as well. The correspondences between values from the different attributes are not lost, therefore the validation function works with n m -tuples as:

$$f_{ch}: D^{n+1} \rightarrow S \quad (2)$$

$$f_{ch}((c_1^1, \dots, c_1^m), (c_2^1, \dots, c_2^m), \dots, (c_n^1, \dots, c_n^m), (c^1, \dots, c^m)) = s, s \in S, c_i^j \in D, \forall i = 1 \dots n$$

For a single attribute conflict handling function, additional information (e.g. one more attribute of ID) can be given as a separate parameter A to unify the single-attribute problem into a multi-attribute one. It can be expressed as:

$$f_{ch}: D^{n+1} \times A \rightarrow S \quad (3)$$

$$f_{ch}((c_1, A_1) \dots, (c_n, A_n), c) = s, s \in S, c_i \in D, \forall i = 1 \dots n$$

The number n follows the number of external data sources. Taking the simplified single-attribute strategy as an example, if $\exists c_i = c, \forall i = 1 \dots n$, then there is no conflict and the function should be evaluated as $f_{ch}(c_1, \dots, c_n, c) = c$. So the validation state of the entity should be "OK". Otherwise a c_i should be selected by some selection rules and be given as a confliction solution to correct attributes of the entity.

2.2.2 Levenshtein distance

As mentioned before, there can be several ways (models) to select tuples with the "highest information quality" as recommended solutions to conflictions. For the underlying quality model, the similarity model is chosen for CytomicsDB. That is, the similarity between the entry from external data source and the entry provided by researchers is viewed as an indicator of the quality. The entries which are most similar to the entry provided by researchers are chosen as recommended solutions to deal with conflictions in attributes. To compare the similarity between two entries, first a similarity score is calculated for each attribute by comparing fact values for the attribute separately. Then a multi-objective decision algorithm gives an overall score on the similarity by considering similarity scores on all attributes.

The types of attributes can be varied a lot from different datasets. However, attributes still can be categorized by their types. For most of the cases, they can be numerical, strings, categorical or taxonomical attributes [5]. For the metadata sets to be validated in CytomicsDB, their attributes can be viewed as strings (digit sequences). Then the task of computing the similarity for a pair of fact values is simplified to grading the similarity between two strings. The score of similarity is represented as the edit distance (or so called "Levenshtein distance" [6]) between a pair of fact values. The Levenshtein distance is a sensitive measure with which distances between strings are calculated. The algorithm finds the cost of the least expensive set of insertions (add a character to the string), deletions (delete a character from the string) or substitutions (replace a character from the one string by a character of the other string) that would be needed to transform one string into the other [7]. The distance between two strings is normalized to [0, 1] range to describe the similarity between these two strings. To compare the similarity between fact strings b and a The Levenshtein distance can be defined recursively as following:

(4)

$$d_{i0} = \sum_{k=1}^i w_{del}(b_k), \quad 1 \leq i \leq m, m = length(b)$$

$$d_{0j} = \sum_{k=1}^j w_{ins}(a_k), \quad 1 \leq j \leq n, n = length(a)$$

$$d_{ij} = \begin{cases} d_{i-1,j-1} & a_j = b_i \quad 1 \leq i \leq m, 1 \leq j \leq n, \\ \min \begin{cases} d_{i-1,j} + w_{del}(b_i) \\ d_{i,j-1} + w_{ins}(a_j) \\ d_{i-1,j-1} + w_{sub}(a_j, b_i) \end{cases} & a_j \neq b_i, \quad m = length(a), n = length(b) \end{cases}$$

In the formula, w_{del} , w_{ins} and w_{sub} represent the weighted function to calculate the cost of deletions, insertions or substitutions. The d_{mn} is the final similarity score between a and b . d_{mn} can be normalized as: $d_{mn}' = 1 - arctan(d_{mn}) \times \frac{2}{\pi}$. This recurrence can be computed as a matrix. An example of computing the similarity score of siRNA sequences "GAATC" and "GATC" (the two sequences which are faked only for example are captured and cut off from the 'siRNADB' [8]) is given here. First, the two sequences are initialized in a matrix as shown in Table 1:

Table 1: initialization of Levenshtein distance

		G	A	A	T	C
	0	1	2	3	4	5
G	1					
A	2					
T	3					
C	4					

The number in each block in the table means the distance score. In the example, all weight numbers for all 3 actions are to 1. The second step is filling the rest blocks following the rule that: 1) if the two corresponded characters for the block are the same (as shown in Table 2, the two characters for block a_{11} are the same "G"), then fill the block with the minimum number in its top-left block; otherwise, fill the block with the minimum weighted number calculated from the numbers added 1 in its left, top and top-left blocks (e.g. the value for block a_{12} in Table 2).

Table 2: fill in the table to calculate Levenshtein distance

		G	A	A	T	C
	0	1	2	3	4	5
G	1	$a_{11} = 0$	$a_{12} = 0+1$			
A	2					
T	3					
C	4					

As shown in Table 3, by repeating above steps until the table is full-filled, the value in the most right-bottom block is then the final similarity score between the two sequences (still needs to be normalized of course).

Table 3: full-filled table for calculating Levenshtein distance

		G	A	A	T	C
--	--	----------	----------	----------	----------	----------

	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	0	1	2	3
T	3	2	1	1	1	2
C	4	3	2	2	2	1

As one can see, the final Levenshtein distance between these two sequences is 1 which means deleting one “A” from “GAATC” sequence will make the two sequences identical to each other.

2.2.3 Multi-objective Decision

As the concern of performance, the multi-objective decision strategy is designed as simple as possible In CytomicsDB. Since the user has the final word on the solutions, the drawback (inaccuracy) of auto-generating possible solutions can be effectively overcome. There are several categories of multi-objective decision (Weighted Global Criterion Methods [9], Analytic Hierarchy Processes [10], Evolution Algorithms, etc.) but the most intuitive way is to convert the multi-objective decision problem to a single-objective decision problem by Weighted Global Criterion Method. Then it only needs to select the biggest sum-scored ones as the recommended solutions. For the entity E described in m domains, assuming n m -tuple entries from external data sources are given as candidates for potential solutions, if the similarity to E_i ($1 \leq i \leq m$) has been scored as s_{ij} ($1 \leq i \leq n, 1 \leq j \leq m$) for each fact-value in the n m -tuple entries, the strategy can be represented as a weighted exponential sum formula:

$$U_i = \sum_{j=1}^m w_j \times (s_{ij})^p \quad 1 \leq i \leq n, 1 \leq j \leq m \quad (5)$$

Then the i^{th} entry with $\max(U_i)$ in the n candidates is going to be given as a solution to conflicts.

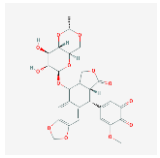
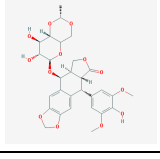
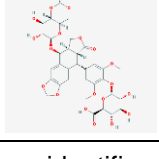
3 Validation Subjects

There are usually two types of data inconsistency [4]: contradictions and duplications. For the contradictions, they may be caused by typos, version updates, shuffle of attributes, etc. Perceiving duplications for entities with unique identities is easily. The perceiving can be performed on the identifier attributes. Otherwise, additional identifier attributes are needed to perceive duplications, which add complexity to the problem. The goal of the validation process applied in CytomicsDB is to detect and correct the inconsistent data in the entities of metadata. In CytomicsDB, metadata attributes are mapped as fields in each table (or so called the “master table”). All the metadata stored in CytomicsDB can be validated for internal consistency to increase the accuracy and reliability of metadata for HTS Experiments. In this thesis, Compounds (which have only one attribute without unique identifier) and siRNAs (which have several attributes beside unique identities) are treated as test cases to implement the validation process.

3.1 Treatment/Compounds

The Treatment/Compound is the most important factor of the experiment. In CytomicsDB, only the name of each Treatment is adopted. The consistency of this treatment name can be validated by using NCBI PubChem Compound database [11]. Here is an example of validating treatment “ETOPOSIDE”. The researchers just offer the name of a compound. The validation process needs to check if the compound’s real name is “ETOPOSIDE” and if the compound has been registered in the master table by another name. By checking the compound with the given name in the NCBI database, the following result is shown in Table 4.

Table 4: The query result by checking the compound name in NCBI database

CID	Name	Name Type	Molecular Weight	Molecular Formula	2D structure
71316630	Etoposide o-Quinone	synonym	572.514120	C ₂₈ H ₂₈ O ₁₃	
	Etoposide 3',4'-Quinone	synonym			
59360017	Etoposide	MeSHHeading	588.556580	C ₂₉ H ₃₂ O ₁₃	
	Etoposide	synonym			
46173784	Etoposide glucuronide	synonym	764.680700	C ₃₅ H ₄₀ O ₁₉	
	Etoposide glucuronide	MeSHHeading			
	Etoposide glucuronide	MeSHTerm			

In Table 4 one is possible to see that the name for the compound is not a unique identifier. A compound entity can be described with several kinds of names like the source name, the Medical Subject Headings (MeSH) and terms, the synonym names, etc. A compound entity can have multiple names in each category as well (e.g. “71316630” compound has two synonym names). Each of these names can be the same as the one from researchers or contains it. Fuzzy search (e.g. searching for “ETQPOSJDE” but “ETOPOSIDE” is got.) is not supported by the NCBI database.

The CID (PubChem Compound Identification) is a non-zero integer PubChem accession identifier for a unique chemical structure. So it can be used as additional information to detect duplications.

As shown in Table 4, the molecular weight and formula is the most distinguishable attributes for the researchers. So these two attributes are included into the process to assist the researchers to take a final decision. The 2D structure of each entity is also used in the similar way.

3.2 siRNA

Small interfering RNA (siRNA) [13] is a class of 20-25 base pairs in length, double-stranded RNA molecules. In common cases, the siRNA is designed as a gene knockdown tool to interfere with the expression of specific genes with complementary nucleotide

sequences. siRNA inhibits expression from its homologous gene (i.e. the sequence of siRNA is a sub-sequence of its homologous DNA's sequence). The symbol, ID, accession number and GI number of the siRNA follows the homologous gene as well. Usually one strand in the double strands of a siRNA sequence is recorded in the database. The sequence of a siRNA talked in the rest of the thesis is a one strand sequence if not specified.

For the HTS experiment, the siRNA target is of crucial importance. One example of a siRNA provided by the researchers is listed below in Table 5.

Table 5: one siRNA example provided by the researchers

Duplex Number	Gene ID	Gene Symbol	Accession Number	GI Number	Sequence
D-004105-01	7272	TTK	NM_003318	34303964	XXX (not disclosed)

The consistency of this siRNA can be validated using external databases like NCBI Nucleotide [12], HGNC (HUGO Gene Nomenclature Committee) Gene symbols/IDs database [14] and BLAST+ (Basic Local Alignment Search Tool) sequence alignment application [15]. By searching with every attribute value (except the Duplex Number which is an internal unique identifier attribute in the research group) of the siRNA as a keyword in the external database, the following result is obtained (Table 6):

Table 6: the query results from all external data sources

Query Keyword	External Data Source	Gene ID	Gene Symbol	Accession Number	GI Number	Sequence
GeneID: 7272	HGNC IDs	7272	TTK	NM_003318	262399359	100% Aligned
GeneSymbol:TTK	HGNC Symbols	7272	TTK	NM_003318	262399359	100% Aligned
Accession Number: NM_003318	NCBI Nucleotide	7272	TTK	NM_003318.4	262399359	100% Aligned
GINumber: 34303964	NCBI Nucleotide	7272	TTK	NM_003318.3	34303964	100% Aligned
Sequence: *****	BLAST+	100969041	TTK	XM_008969441.1	675737708	100% Aligned
		7272	TTK	NM_003318.4	262399359	100% Aligned
		7272	TTK	NM_001166691.1	262399360	100% Aligned

Table 6 shows that querying with each fact value of attributes in the siRNA from researchers in external data sources may get different results. For example, as querying

with the fact value “*NM_003318*” of Accession Number attribute in NCBI Nucleotide database, the result entry of siRNA has a different GI number (“262399359”) to (“34303964”) the one provided by researchers. It is because the siRNA has a new version (GI Number “34303964” corresponds to Accession Number “*NM_003318.3*” which means the third version of the siRNA while GI Number “262399359” corresponds to Accession Number “*NM_003318.4*” which is the 4th version of the siRNA) stored in the NCBI Nucleotide database. Conflication like this or other typos (e.g. the gene symbol given by the researchers might be miss-spelled or be using a synonym name) will be detected and presented to the user along with best matches from the external data sources and ask the user if he wants to use his own or one of the best matches.

It is possible to notice that the result of query with some fact value from non-unique attribute may get non-unique results (e.g. searching the short sequence against BLAST+ application). Searching with Gene ID, Gene symbol and sequence in external data sources all may get multiple results. That’s one of the reasons that similarity measure and multi-objective decision are highly needed for automatically determination.

To detect the duplication of all the siRNAs in master table, only the attribute “*duplex number*” is used as it is an internal unique identifier attribute.

4 the Validation Workflow

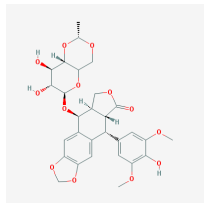
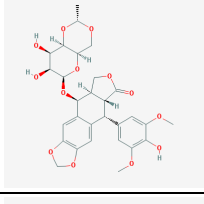
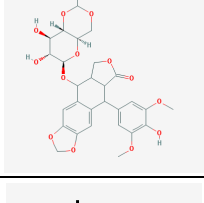
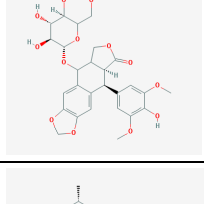
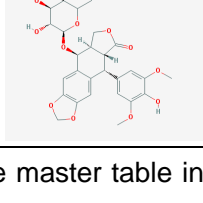
The validation workflow follows “*Trust Your Friends*” and “*Pass It On*” strategies while using “*Levenshtein distance*” and “*Multi-Objective Decision*” algorithms. There are two branches separately focusing on single-attribute and multi-attributes situations in the validation workflow. The two branches follow a common principle of the validation workflow. The principle is first parsing each fact value (the attribute value of metadata from researchers) into a standard unique identifier value by querying it as a keyword in an external data source, then getting the entries from an external database (which should be reliable) which uses the unique identifier as a primary key. The validation of Compounds and the validation of siRNAs can be viewed as two scenarios corresponding to the two branches of the workflow, respectively.

4.1 Treatment/Compounds Validation

Validation of a Compound name is the case for validating single-attribute entities. Before inserting compound names into the master table, a syntactic validation will check if there are duplicated names already registered in the master table. Then later validation process will check the internal consistency of the entry inserted into master table. As mentioned before, since the attribute “*Treatment Name*” of a compound is not a unique identifier, the first step is parsing each name into the unique identifier, “*CID*”, by using external data sources. So the compound name retrieved from the master table is pushed to the parsing stage. Two soap based web services [16] are used to do query in NCBI PubChem Compound database. Using “Esearch” service[19] to query one compound name will get a

list of candidate “CID”s. Delivering the list of “CID”s to “Eesummary” web service[19] will get a list of corresponded compound entries (with attributes like molecular weights, molecular formulas, URLs to 2D structure images, etc.). The list of entries will be candidates for similarity comparing and screening. Entries which have the highest similarity scores among all candidates are picked up as the validation result. These entries are delivered to the user for decision. If there is any inconsistency in the compound, then these entries are potential solutions to them. An example of validation result which are about to “pass on” to users for final decisions is shown in Table 7.

Table 7: a scope on treatment_validation_details table (example)

Treald	treaName	Result Id	CID	nameType	Molecular Weight	Molecular Formula	2DStructure
1714	VP 16-213	1	59360017	synonym	588.556580	C ₂₉ H ₃₂ O ₁₃	
1714	VP 16-213	2	50989217	synonym	588.556580	C ₂₉ H ₃₂ O ₁₃	
1714	VP 16-213	3	11758093	MeSHTerm	588.556580	C ₂₉ H ₃₂ O ₁₃	
2503	Etoposide	4	45356822	synonym	588.556580	C ₂₉ H ₃₂ O ₁₃	
2503	Etoposide	5	59360017	synonym	588.556580	C ₂₉ H ₃₂ O ₁₃	

These example records are stored in a validation result table beside the master table in the database in CytomicsDB. The “treaName” field is the fact value of compound name from researchers. The “treald” field is an auto-generated primary identifier in the master table. The “CID”, “MolecularWeight”, “MolecularFormula” and “2DStructure” fields are fact values of entries retrieved from NCBI PubChem Compound database. The compound

name for each of these entries in the table is perfect matched (according to similarity comparison) to the "treeName" from researchers. So these entries' names are not listed in the table. The "nameType" field indicates the type of these entries' names. The "resultId" filed is an auto generated primary key value in the validation result table.

It is mentioned in chapter 3.1 that query with a compound name (which may be matched at any part in different types of names in external data sources) will get 1-to-many corresponded "CID"s which will lead to multiple entities from external database. As one can see, The query in NCBI PubChem Compound database with compound name "VP 16-213" (and also for compound name "Etoposide") gets not only one perfect matched entries. These entries have the same molecular weight and molecular formula. Only some tiny differences on structures (location of Hydrogen bonds) are distinguishable for them. Besides that, one entry identified (CID) as "59360017" is hit by both names in query.

So a strategy should be applied here to narrow down the choices space and to give potential solutions considering possible duplex. The basic idea is that if only one unregistered (by other names in the master table) perfect matched compound is found in the external data source whose CID is not registered by other entries in the validation result table, then the validation state of the given name is OK. Otherwise if multiple perfect matched compounds are found in the external data source, then the lead researcher (commonly is the administrator of the platform who might be the team leader in the lab) should decide on which one is exactly the "real" compound matched to the given name. Given the compound name "Etoposide" as an example, the user will be informed about multiple hits and a duplex while browsing No. "59360017" compound as an additional warning. Upon the decision made by the researcher, the result should be updated in the validation result table. If one unduplicated candidate ("45356822") compound is confirmed as the only match to "Etoposide", then other candidates (here is only the "59360017" compound for "Etoposide") should be removed from the validation result table. Meanwhile, when the user reviews candidates for "VP 16-213", there should no more duplication warning for "59360017" compound. Otherwise, if the user ignores the warning and confirms that No. "59360017" compound is the only match for "Etoposide", then the decision should be checked in the validation result table. If "59360017" compound has been decided as the only match for name "VP 16-213" which logically means that "Etoposide" and "VP 16-213" are duplicated to each other, then the name "Etoposide" should be removed from master table along with all its validation results. If the corresponding compound for "VP 16-213" is not decided yet, then while deleting another candidate No. "45356822" for "Etoposide", the state of candidate No. "59360017" for "VP 16-213" should be updated to "duplex" which will lead to a duplex warning message when the user review the candidate to make the decision for "VP 16-213". One occasion is that only one perfect matched compound is found in NCBI database for the given name but this compound is duplicated in the validation result table. In this situation, the matched

compound will be inserted into the validation result table with a duplex state and user should make a decision on the candidate. The confirmation process is the same as talked before in this paragraph. Assuming that the list of candidates (the list is not empty) for Compound Name n has been retrieved as l , the strategy is expressed as the following pseudo code:

Start

Step1: get best matched candidates c from l

Step2: if $length(c) = 1$ && $duplexDetect(c_0) = false$ → return Validation State = 'OK',

else → go to Step3

Step3: for $i \leftarrow c_0$ to $c_{length(c)-1}$

if $duplexdetect(i) = true$ → set $i.state = 'duplex'$

else → $i.state = ''$

end for

go to Step 4

Step4: wait for the user's decision among c

(abort for users interaction)

Step5: get the user's decided candidate d among c

Step6: if $d.state = 'duplex'$ → go to Step7, else → go to Step9

Step7: get entries list el by $d.CID$ from the validation result table

Step8: for $e \leftarrow el_0$ to $el_{length(el)-1}$

if e is waiting for users' decision → set $e.state = 'duplex'$

else → delete the e 's corresponded treatment name in master table;

delete e

end for

go to Step 9

Step9: get entries list el with $CIDs$ which are among all candidates of n except d from the validation result table

Step10: for $e \leftarrow el_0$ to $el_{length(el)-1}$

if $e.state = 'duplex'$ → set $e.state = ''$

end for

go to Step11

Step 11: delete other candidates except d for n

end

Going back to the example, in fact the two names are all synonyms to the compound identified as "59360017". So they are theoretically duplicated to each other. If the user makes a professional choice, one of the two treatment name should be removed from the master table following the duplex detection process.

For a compound name which fails to get perfect matches among the query results from NCBI database, the top 3 "best matched" entries will be given as potential solutions to correct possible typos in the compound name. The entries "passed on" to users follow the same scenario (as talked in the last paragraphs) to warn researchers about duplexes and to update internal database according to users' choices.

The workflow is used while the researcher inserting, updating or deleting a compound name from the master table. So some common functions like “delete entries from validation result table” and “mark duplex” are implemented as components for reusing. Based on these components, the whole workflow can be divided as four stages, i.e. “getting candidates”, “screening & marking duplex”, “updating duplex marks” and “cleaning up the validation result table”. The following diagram shows the workflow in each components and how they interact with each other.

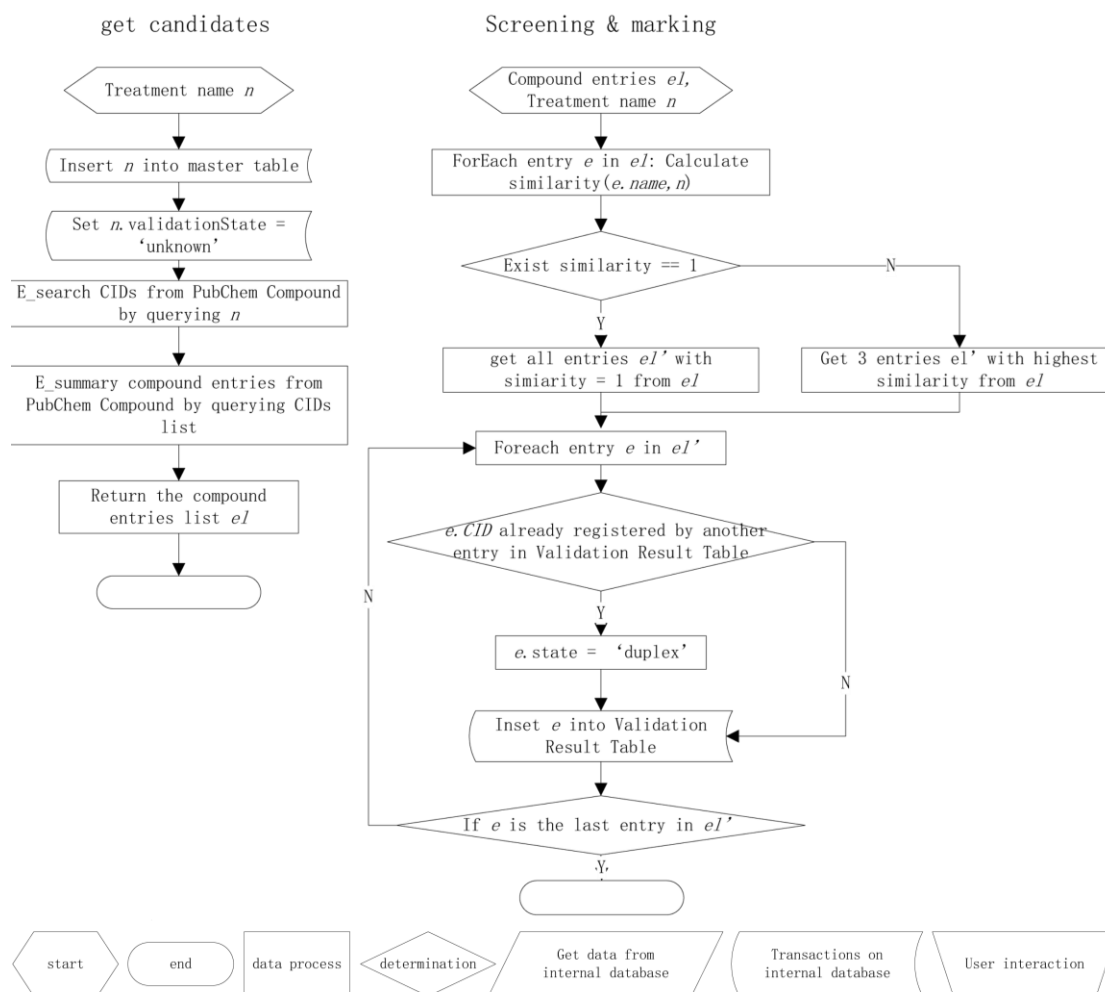


Fig. 2 (1) the workflow of "get Candidates" and "Screening & marking" components

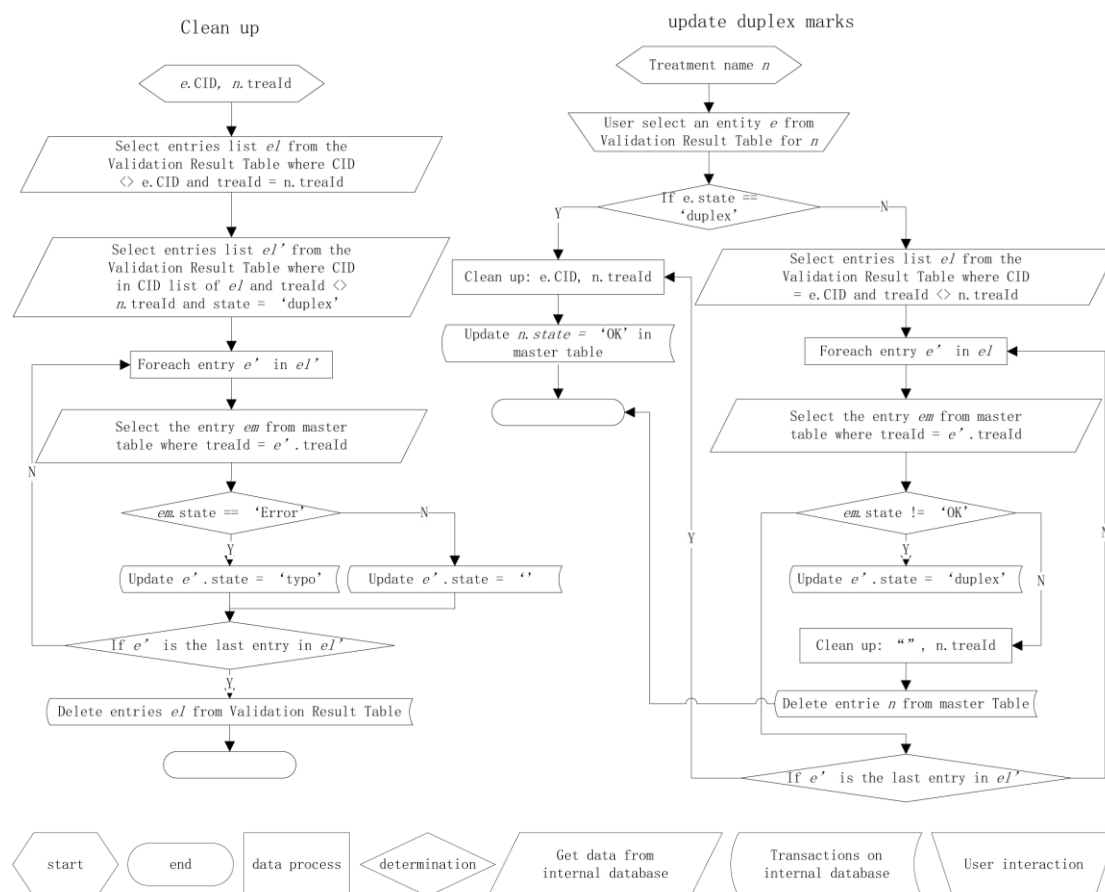


Fig. 2 (2) the workflow of "Clean Up" and "Update Duplex Marks" components in treatment validation

While the user wants to insert or update a compound into the master table, the process starts to check whether the name has existed in the master table. If so, the name will not be stored. Otherwise, the process will call components in the order, i.e. "get candidates"->"screening"->"update duplex marks". If the user wants to ignore the validation results and keep the compound in the master table as it is, the user can select "ignore" which will call "clean up" component. If the user wants to delete the compound from master table, then after calling "clear up" component, the deleting action in the master table will follow.

4.2 siRNAs Validation

This is the case for validating multi-attributes entities. 5 types of siRNA attributes can be validated with external data sources. They are: the Gene ID, the Gene Symbol, the Accession Number, the GI number and the sequence. Since the Duplex Number (it is only used on the master table as a unique identifier which cannot be validated with external data source) attribute is registered on each siRNA entry in the master table, only a basic duplex validation of siRNAs is adopted, i.e. assuming the Duplex Number of the user-provided siRNA is reliable, then only checking if the duplex number has existed in the master table is enough.

To do the validation, not all the 5 kinds of attribute values can be directly used as input

fields for query in external data sources. Meanwhile, not all the 5 attributes are included in the query output for each external data source. The supported types of attributes for query input/output in external data sources are listed below in Table 8.

Table 8: supported types of attributes (input & output) in external data sources

External Data Source	Supported types of attributes for query	Query Output fields
HGNC IDs	GeneID	GeneID, GeneSymbol, Accession Number
HGNC Symbols	GeneSymbol	GeneID, GeneSymbol, Accession Number
NCBI Nucleotide	Accession Number, GINumber	Accession Number, GINumber, GeneID, GeneSymbol, sequence
BLAST+	Accession Number, GINumber, Sequence	Accession Number, GINumber, sequence

As shown in Table 8, each data source has some specified input fields (e.g. only BLAST+ application can search with a sequence and only HGNC databases can search with Gene ID/ symbol, etc.). And not all five fields are available in the output siRNA entities (in fact only NCBI Nucleotide database support all fields in the output). However, all these data sources do have a common field, “Accession number”, in the output. The Accession Number (or so called “GenBank Accession Number”) is a unique identifier given to a DNA entity record to track versions and associated entities over time of the entity record in a data repository [17]. A standard example of an accession number in table 6 is “NM_003318.4”. [18] It is a combination of an accession prefix (“NM_003318”) and a version number (“4”). If the sequence of the DNA entity changes, the accession prefix will remain the same but the version number will increment. GenBank GI number, however, will change each time the sequence changes – even if only one base is affected. So the accession number is used as a common identifier in the validation process.

Another issue needs to be addressed is that the RNA sequence returned from external data sources is in fact a single strand (from the original two strands of the RNA) which can be viewed as mRNA. Although the homologous siRNA's sequence provided by users is classified which is not authorized to be used in this thesis, just as described in chapter 3.2, one can imagine that the sequence of the siRNA should be possibly aligned perfectly to a part in its homologous mRNA strand (the “T” base and “U” base are equivalent for BLAST+ application in alignment). The BLAST+ application offers a functionality to align sequences and give a similarity score of the two sequences after alignment.

An example of a user-provided siRNA is given in table 9. Attribute values from “Order Number”, “Pool Catalog Number” and “Duplex Number” are not possible to be validated with external data sources as they are only defined and used internally in the lab. So they are not considered in the validation process. In Table 9, these fields are grey-marked.

Table 9: an example for a user uploaded siRNA

Gene ID	7272
Gene Symbol	TTK
Order Number	191376
Pool Catalog Number	D-004105-01
Accession Number	NM_003318
GI Number	34303964
Duplex Number	D-004105-01
Sequence	*****

In the first step to validate this siRNA, each attribute in the entity is parsed by a separate parser. The multi-attributes validation problem is then transformed into a single-attribute validation problem. Following the scenario talked in chapter 4.1, the first step of the validation process is parsing each attribute into the unique identifier, Accession Number, by using external data sources. The resolution of returned Accession numbers varies from databases to databases according to their settings. For example the accession number returned from HGNC databases omits the version number suffix. For a given Accession number without suffix queried in NCBI Nucleotide database (i.e. query with “NM_003318”), the returned accession number is always the latest version one. The Accession number from the user is usually without version suffix. So in the first stage, only the accession prefix of hit accession number is collected. The example result of first stage is listed in Table 10.

Table 10: the list of accession prefix generated from the first stage of siRNA validation

Input field & value	Parsed to	External data source
Gene ID: 7272	NM_003318	HGNC Gene ID
Gene Symbol: TTK	NM_003318	HGNC Gene Symbol
Accession Number: NM_003318	NM_003318	NCBI Nucleotide
GI Number: 34303964	NM_003318	NCBI Nucleotide
Sequence: *****	XM_008969441	BLAST+
	NM_003318	
	NM_001166691	

The duplicated results (as those grey cells shown in Table 10) are omitted from the list before sending the list to next step. As the example given in chapter 3.2, it is a common sense (or an empirical assumption) that if there is some inconsistency between the siRNA entry provided by the user and the siRNA entry found in external data source, it's highly possible because of the version update. This means there is a comparatively higher chance to find a perfect match in one of the versions of the siRNA. So the next step is finding all existed version suffixes for accession numbers retrieved from the first step. These accession numbers (with version suffixes) corresponded RNAs are used as candidates for the next step. The latest version of accession numbers are fetched from NCBI Nucleotide database by NCBI Eutilities EFetch web service [19]. An iterator then is

applied on the version suffix to get a list of accession numbers from version one to the latest one. For those accession prefixes in step 1, the list of corresponded accession numbers is put in Table 11.

Table 11: the list of accession number generated from the second stage of siRNA validation

Accession numbers	Latest version
NM_003318.1	4
NM_003318.2	
NM_003318.3	
NM_003318.4	
XM_008969441.1	1
NM_001166691.1	1

Then, RNA candidates from NCBI Nucleotide database by querying with these accession numbers using the NCBI EFetch service. The RNA candidates are listed in Table 12.

Table 12: the list of siRNA candidates from NCBI database

Acc-numbers	GI numbers	Gene ID	Gene Symbol	Sequence
NM_003318.1	4507718	7272	TTK;MPS1L1	Appendix link.1
NM_003318.2	23308721	7272	TTK;ESK;MPS1L1;PYT	Appendix link.2
NM_003318.3	34303964	7272	TTK;CT96;ESK;FLJ38280;MPS1;MPS1L1;PYT	Appendix link.3
NM_003318.4	262399359	7272	TTK; CT96;ESK;MPH1;MPS1;MPS1L1;PYT	Appendix link.4
XM_008969441.1	675737708	100969041	TTK	Appendix link.5
NM_001166691.1	262399360	7272	TTK;CT96;ESK;MPH1;MPS1;MPS1L1;PYT	Appendix link.6

One may noticed that, some candidates have several names in Gene Symbol attribute (e.g. “TTK; ESK; MPS1L1; PYT” for the “NM_003318.2” entry). It is because except one official gene symbol (here is “TTK”), one siRNA can have several synonym names. While retrieving candidates, all these synonym names will be collected and put behind the official gene symbol in the Gene Symbol field. Then if the siRNA from the user uses a synonym name, the validation process can detect it and give a warning to the user.

The fourth step is using several comparators to calculate the similarity of each attribute between the candidate and the siRNA from the researcher. The result of this step is a matrix of size $n \times 5$ where n is the number of candidates. Each row in this matrix corresponds to an entry of candidates and each column corresponds to each attribute. The similarity score is stored in each cell of the matrix. Given one candidate RNA s_c , the siRNA s from the researcher and the Levenshtein distance similarity grading function $ld(attr1: string, attr2: string)$, the comparators used in this stage are listed below.

Gene ID & GI number comparator: It simply calls the Levenshtein distance function to compare the similarity of attribute values of s_c and s . To calculate the similarity score of Gene ID values, the function $ld(s.geneld, s_c.gengld)$ is used. The similarity score of the

two GI Number values equals to $ld(s.GI\text{Number}, s_c.GI\text{Number})$.

Gene Symbol comparator: The result of this comparator is a tuple because it is crucial to calculate the similarity score between s 's gene symbol and s_c 's official gene symbol separately from the score between s 's gene symbol and s_c 's gene synonym. The tuple can be presented as:

$$\left\{ \begin{array}{l} ld(s.GeneSymbol, s_c.GeneSymbol.split(':')[0]) \\ \max(i: \text{From } 1 \text{ to } n, ld(s.GeneSymbol, s_c.GeneSymbol.split(':')[i])) \end{array} \right\}$$

where n equals to $s_c.GeneSymbol.split(":",").length-1$.

Accession number comparator: An accession number needs to be compared in two separated parts. As the accession number from the user usually does not have a version suffix, the comparison between suffixes is meaningless. So the comparing of accession numbers focus on the accession prefix part. In the prefix, there is a 2-letters-start (i.e. "NM" in "NM_003318.4") followed by several numerical digits (i.e. "003318" in "NM_003318.4"). The two parts are separated by a "_". The 2-letters-start presents the species the gene bellows to. The comparator returns the minimum similarity score of the two parts as output. It can be presented as:

$$\begin{aligned} s.AccessNumber &= s.AccessionNumber.split('.') \\ s_c.AccessNumber &= s_c.AccessionNumber.split('.') \\ s.AccessPrx &= s.AccessNumber.split('_')[0] \\ s_c.AccessPrx &= s_c.AccessNumber.split('_')[0] \\ s.AccessMidx &= s.AccessNumber.split('_')[1] \\ s_c.AccessMidx &= s_c.AccessNumber.split('_')[1] \end{aligned}$$

similarity score = $\min(ld(s.AccessPrx, s_c.AccessPrx), ld(s.AccessMidx, s_c.AccessMidx))$

Sequence comparator: The similarity score of two sequences is described by the bit-score and e-value from BLAST+ application. There is a raw score $S = \sum_{i=1}^L s_{r_{1i} r_{2i}}$ which is a numerical value that describes the overall quality of an alignment. Higher numbers correspond to higher similarity. The score scale depends on the scoring system used (substitution matrix, gap penalty). An example of calculating the raw score is shown in Fig. 3.

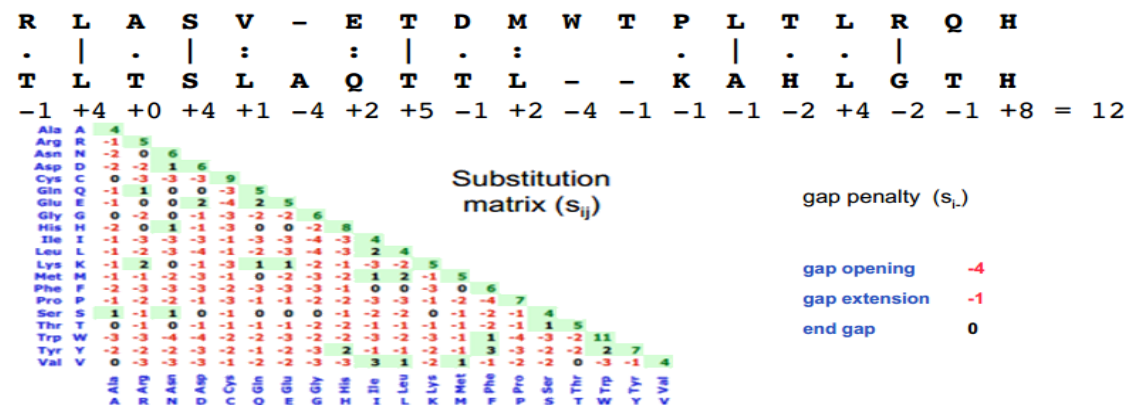


Fig. 3 example of calculating the raw score in BLAST+

In the context of sequence alignments (BLAST), the bit-score S' is a normalized score expressed in bits that shows the estimation on the magnitude of the search space where one would have to look through before he or she would expect to find a score as good as or better than this one by chance. The bit-score follows the following definition:

$$S' = \frac{\lambda S - \ln(K)}{\ln(2)} \quad (6)$$

where S is the raw score. Parameters λ and K depend on the substitution matrix and on the gap penalties. If the bit-score is 30, one would have to score, on average, about $2^{30} \approx 1$ billion independent segment pairs to find a score could match this score by chance. Each additional bit doubles the size of the search space (which is proportional to the product of the query sequence length n multiplying the sum of the lengths of the sequences in the database m). So the size of the search space is obtained by $N=n \times m$). The bit-score is thus a rescaled version of the raw alignment score.

The e-value or so called "Expectation value" is the number of distinct alignments, with a score equivalent to or better than S , that are expected to occur in a database search by chance. The lower the e-value, the more significant the score is.

The comparator calculates the final similarity score as:

$$\text{similarity score} = \arctan(\text{bit-score} / \text{e-value}) \times \frac{2}{\pi} \quad (7)$$

which will lead to a similarity score within range $[0,1]$.

The example of resulted matrix generated from the fourth step is attached in Table 13.

Table 13: the example of similarity matrix

	GI numbers similarity	Gene ID similarity	Gene Symbol similarity	Accession Number similarity	Sequence similarity
siRNA from user	Compare to: 34303964	Compare to: 7272	Compare to: TTK	Compare to: NM_003318	Compare to: User's Sequence
Candidate1	0.0056	1	<1,0>	1	1
Candidate2	0.1051	1	<1,0>	1	1
Candidate3	1	1	<1,0>	1	1
Candidate4	0.0792	1	<1,0>	1	1
Candidate5	0.0792	0.07	<1,0>	0.0056	1
Candidate6	0.0903	1	<1,0>	0.0056	1

The fifth step of the process is using the multi-objective decision method to screen best candidates and feed it back to the user for decision (if a perfect matched candidate is found then the user will get a positive feedback for instead. e.g. the candidate 3 in Table 13 scores 1 for all comparators except the Gene Symbol comparator. For the Gene Symbol comparator, either cell in the returned tuple is 1. It is a perfect match for the siRNA). If no perfect matched candidates are found, then an error message along with the top 3 best matched candidates decided by the multi-objective decision method will be sent

to the user. If the perfect matched candidate is targeted, it still needs to check if a new version of the gene exists in external database (e.g. the candidate 4 in Table 13) or if the candidate is using a synonym name (the value of the first cell in the Gene Symbol comparator returned tuple is less than 1 while the second cell value is 1). If so, a warning message will be sent to the user. The user can choose to ignore solutions from the validation process or accept one as a correction. The workflow of the whole process can be presented in Fig. 4.

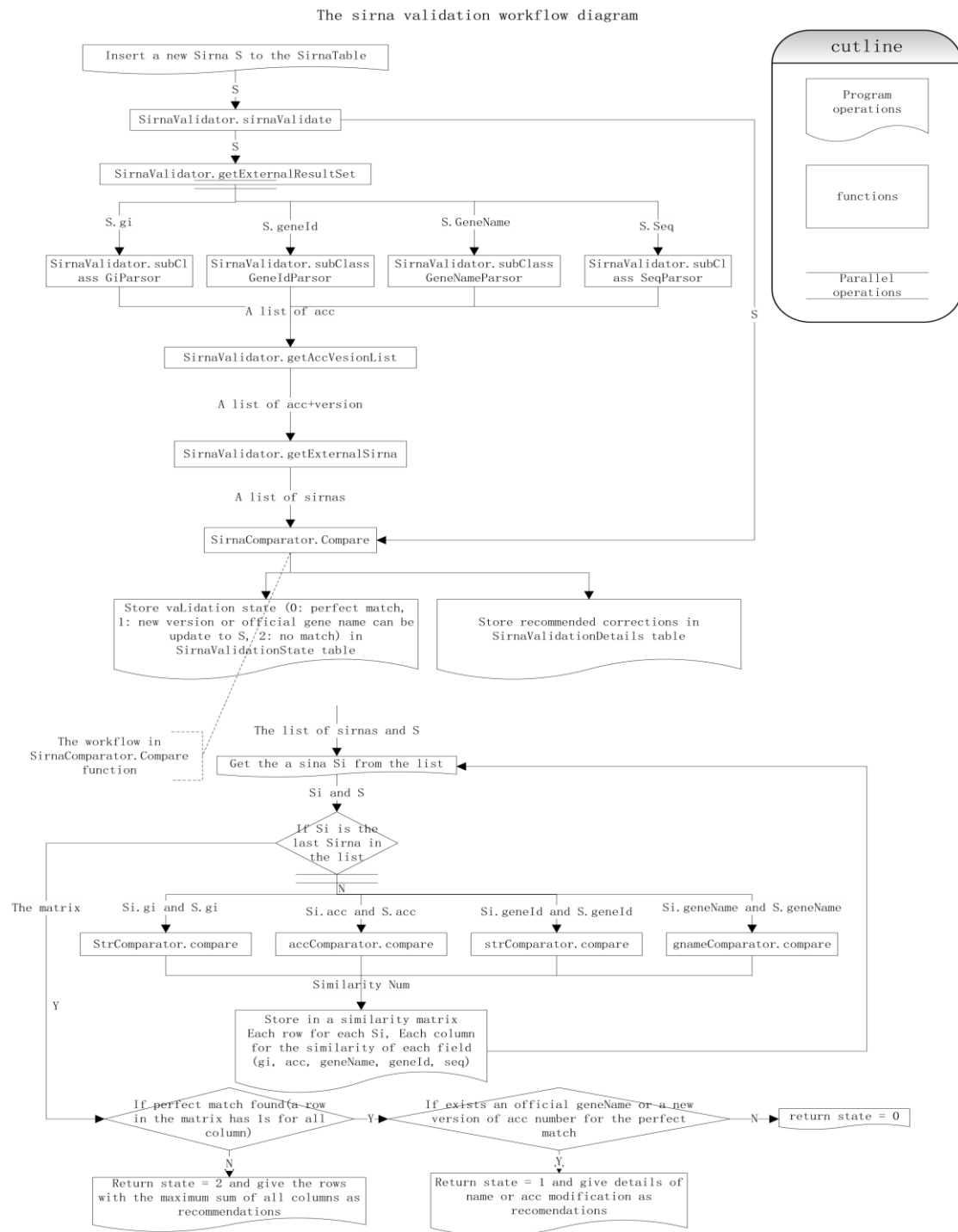


Fig. 4 the workflow of siRNA validation process

Parsers in stage one and comparators in stage four are put to multi-threads to run parallelly. The BLAST+ application itself can be set to run in a parallel way as well. When the user adds a new siRNA to the master table or updates one the validation process will run automatically. A detailed class diagram of this validation process can be checked in Fig. 1 in Appendix.

5 the Architecture

The architecture is designed to support the workflow of validation described in section 4. The performance stability, speed and pressure distribution are main concerns for the architecture because the overhead of connecting to external web services and loading a BLAST+ local sequence database into the ram are heavy tasks during the validation. Besides that, since the workflow relays on external application (e.g. BLAST+ gets different I/O schema in Windows and Linux), it needs to concern the compatibility cross operation systems.

The validation process consists of four main activities: retrieving candidates, screening candidates, reporting inconsistency, and updating master table according to users' decision. The four activities are distributed in several components which interact with each other (The component diagram in Fig. 5 shows the interaction between these components). These components can be categorized into a five-layer architecture which is composed with a presentation layer, a utility layer, a services layer, a persistence layer and a data repository.

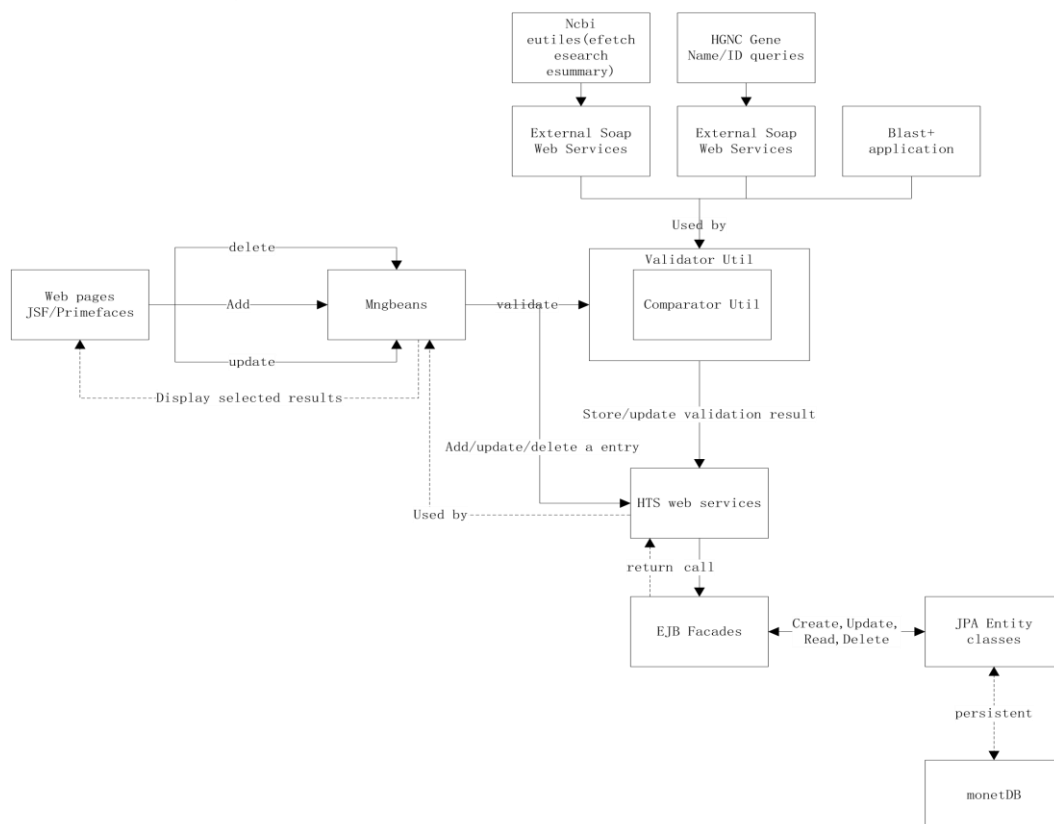


Fig. 5 components diagram of validation workflow

5.1 The presentation layer

The validation process is functionally enabled for users using a single web based graphical user interface. The presentation layer supports the GUI for users. Coded in the JSF 2.0 and PrimeFaces 4.0 front end framework which fully support HTML5 and JavaScript/AJAX, the presentation layer makes it easier for users to interact with data in master tables [20]. On the web page, users can send requests to batch-upload a list of entries into master tables, create or update a single new entry in a master table, view entries in master tables, or delete one entry. The validation process will be triggered by the user's operations on the presentation layer. During uploading or creating entries in the data repository, the validation process will be triggered to validate the new entries at the back end. While the user viewing the detail of one entry, the validation result and recommended solutions show synchronously. Users with privilege can direct make decisions (choose one candidate to correct the detected inconsistency or keep his own one in the master table) on the view dialog. The choice will be updated into the data repository by the validation process. After the user deleting or updating an entry from the master table, the validation process will be called to update the duplication information in the validation result table if necessary. Besides that, the presentation layer is the first stage of validation in the platform by considering mandatory fields for uploading experiment metadata. The presentation layer also controls messages bubbling. Some errors happened during the validation process will be reported to users by an alert message on the web page. Fig 6 is a collection of screenshots of the presentation layer.

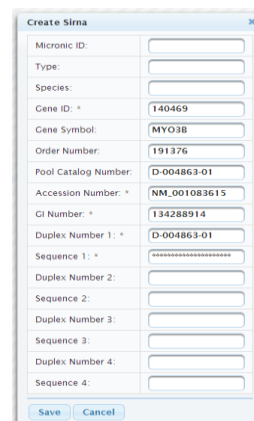
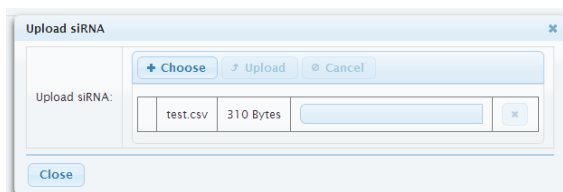


Fig. 6 (1a) upload new entries into the master table **Fig. 6 (1b)** create a new entry in the master table

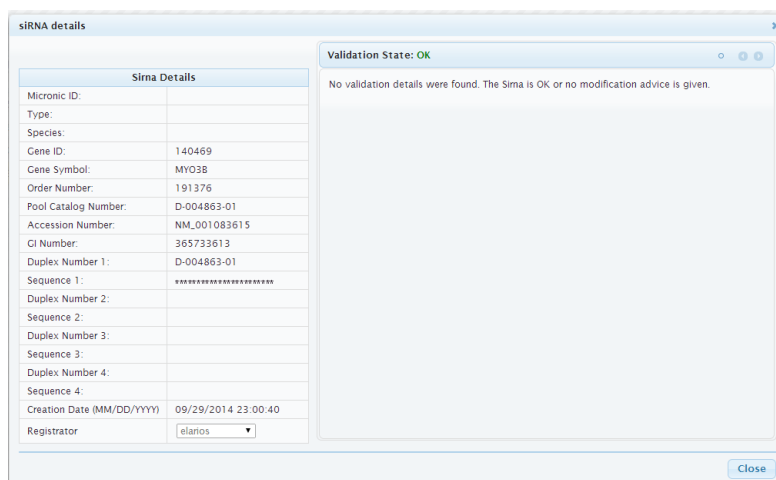


Fig. 6 (2) in the view dialog one of the entries passed the validation

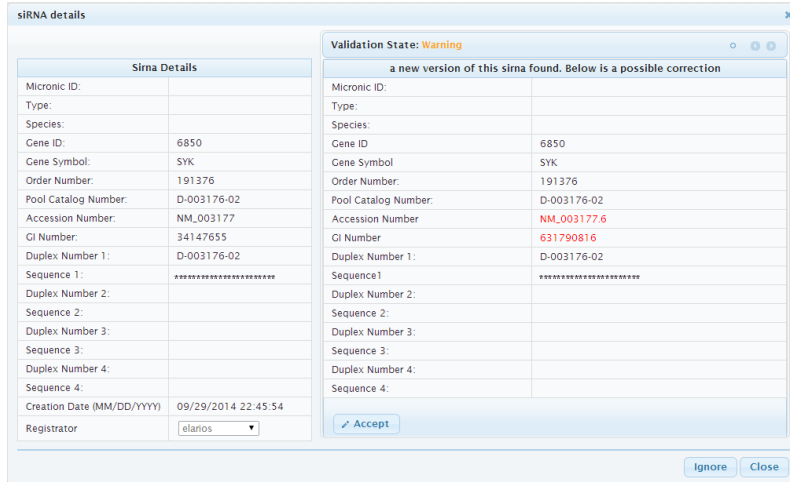


Fig. 6 (3) a new version was found for one entry in external database

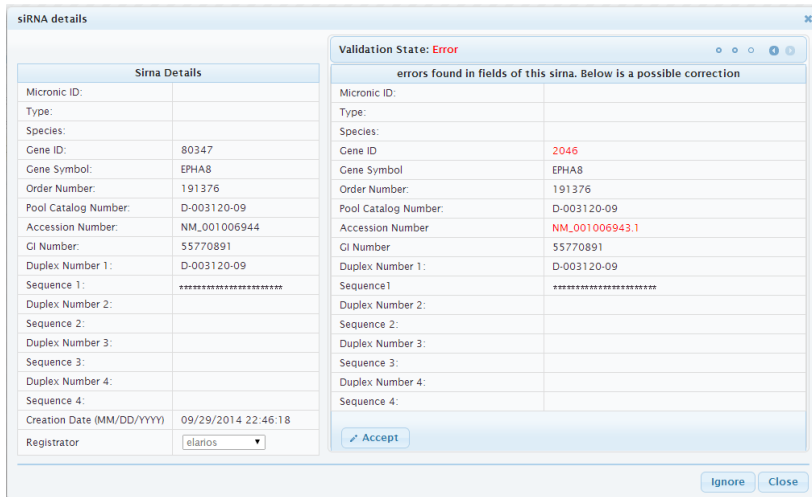


Fig. 6 (4) one entry gets a typo in the last base of Accession Number and used a wrong Gene ID

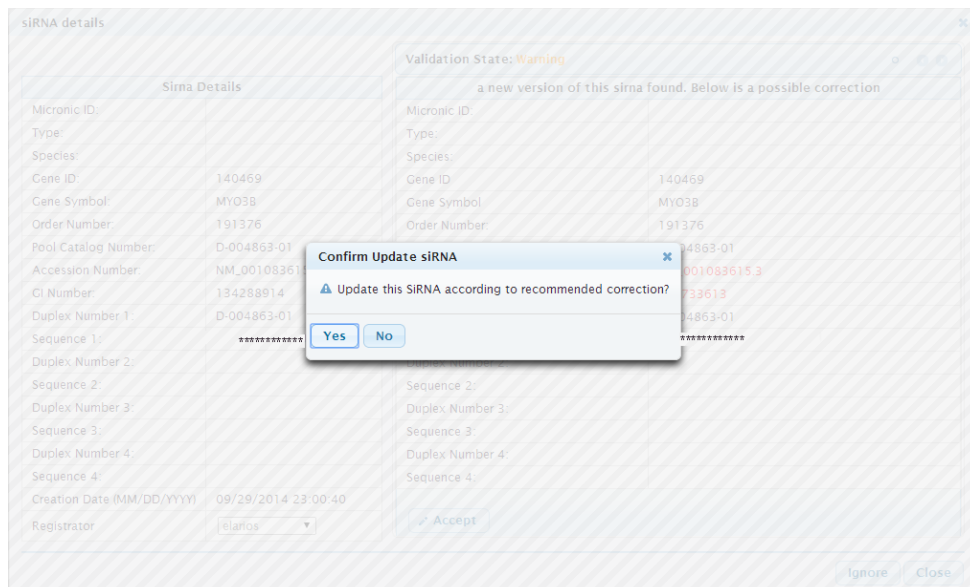


Fig. 6 (5) the administrator can decide to "accept" the correction

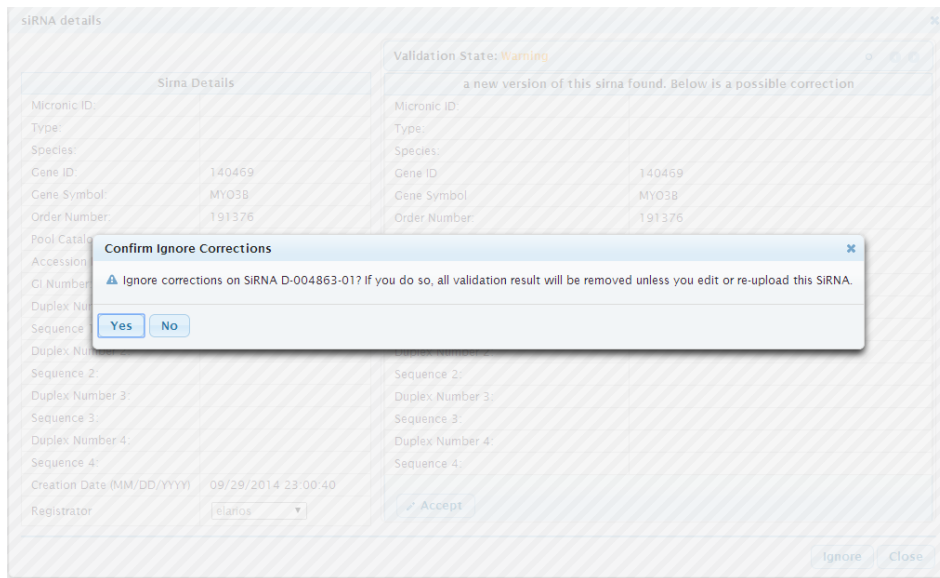


Fig. 6 (6) the administrator can decide to "ignore" the correction

List of siRNAs						
Duplex Number 1	Gene Symbol	Gene ID	Accession Number	GI Number	Creation Date (MM/DD/YYYY)	Validation State
D-004105-01	TTK	7272	NM_003318	34303964	09/29/2014 22:39:19	⚠
D-003176-02	SYK	6850	NM_003177	34147655	09/29/2014 22:45:54	⚠
D-003120-09	EPHA8	80347	NM_001006944	55770891	09/29/2014 22:46:18	✖
D-004863-01	MYO3B	140469	NM_001083615	365733613	09/29/2014 23:00:40	✓

Fig. 6 (7) the validation result can be overviewed in the grid

5.2 The utility layer

The utility layer includes manage beans and several utilities like parsers and comparators. They work as the pivot in the validation process to control the generation of candidates, the screening of candidates and the responding actions after the researcher makes a choice on the presentation layer. The manage beans are controllers to request the utilities to visit resources from external web services (or applications) and do calculations. They also control the calling of internal web services in the service layer to do CRUD (create, update, read and delete) actions in master tables. Running results or errors are collected and sent back to the presentation layer. The purpose of separating those utilities from manage beans (as independent components) is to make it easier to run those utility instances in a parallel way.

5.3 The service layer

This layer consists of multiple web services which support every step in the HTS workflow. These web services invoke different APIs which are in charge of the Experiment design, Image Analysis and Data Analysis [20]. This structure allows easy extension with more functional modules. For example, parsers in the utility layer use web services to access external data sources. The Simple Object Access Protocol (SOAP) messages are selected for invoking the web services and receiving results because of its approved interoperability in web applications and heterogeneous environments. For these web services, one big portion of work is keeping the persistence in the database by using

modules from the persistence layer. The MonetDB (www.monetdb.org) database used in CytomicsDB is not a transaction database [2]. Therefore, operations like insertions, updates and deletions are minimized by batching them as many as possible.

5.4 The persistence Layer

This layer is based on the principle of object-relational mapping (ORM) which involves delegating access to relational database, which in turn give an object-oriented view of relational data, and vice versa. The Java Persistence API (JPA) framework has been implemented in this layer to keep a bidirectional correspondence between the database and objects. Those Java objects used in the framework are known as Java Entities [21]. Entities are objects that live shortly in memory and persistently in the database. Besides that, they have all the features of a Java class like instantiation, abstraction, inheritance, relationships and so on. The entities used in CytomicsDB follow the structure of the tables they mapped to i.e. mapping the fields as properties in the objects. Basic CRUD (create/read/update/delete) operations are registered as named query methods which are written in Java Persistence Query Language (JPQL). Customized queries can be attached to entities as native queries via JPA.

5.5 The repository

The master tables are stored in an open source column-based database system, MonetDB, which is used as the data repository. A validation process needs to use three tables. Their structures are shown in Fig. 7. The tables in the middle (“sirna” table and “treatment” table) are the master tables. The validation state (“OK”, “Warning” or “error”) for each entity in the master table is stored in the validation state table. In the validation details tables, solutions to correct inconsistencies in each entity are stored there. Therefore, the entries in validation state table are one-to-one correspond to those in master tables while the validation details tables have one-to-many correspondence to master tables.

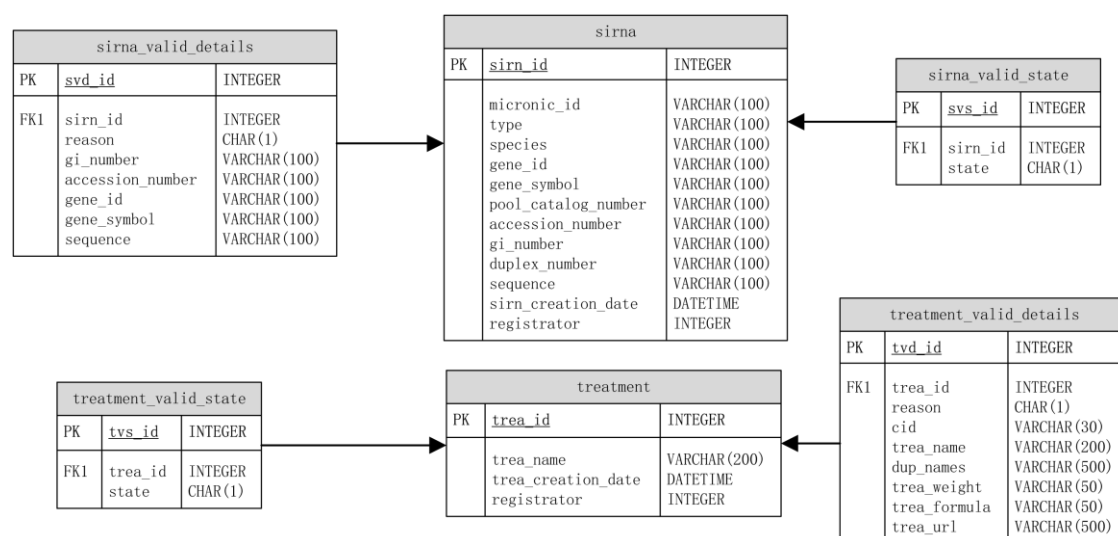


Fig. 7 the database structure

MonetDB is proved to have superior performance in processing analytical queries on large scale data [2] which is suitable for the complex data manipulation in the validation workflow. Thus, in CytomicsDB, MonetDB is used to store the experiment metadata and validation intermediate results.

The overall architecture of modules in CytomicsDB is shown in Fig. 8.

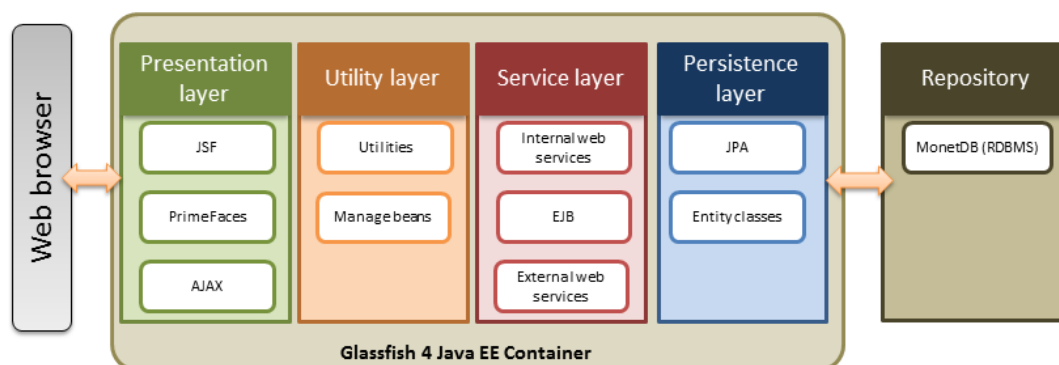


Fig. 8 CytomicsDB architecture

6 Evaluation of the results

This thesis analyses the overall performance of the validation strategy from three aspects. The evaluation focuses more on the accuracy of the results from the validation progress than other factors like the efficiency which could be important for a real-time web based system.

The efficiency is indeed an important factor. Running in a multi-threads pathway with minimized I/O (input/output) to all web services, it will take two seconds to validate one single-attribute entry or four seconds or so to validate a multi-attributes entry in a Linux system with stable Internet connection. However the time overhead is not going to be accurately measured and evaluated in this thesis. The first reason is that this validation process is performed during the experiment prepare stage. Once the metadata uploaded and validated, changes on metadata will be very limited (i.e. “once created, use forever”). So the overhead of one round validation can be a less important issue than its accuracy. The second reason is that the overhead relies on several other aspects like: 1) the speed of the Internet, 2) the implementation of multi-threads in the core architecture of the operation system, 3) the operation system which the BLAST+ application runs on (BLAST+ application requires extra I/O operations with local file system in Windows system). These issues are not going to be discussed elaborately in this thesis. So only the rough estimation of time overhead is given in this thesis.

As talked in chapter 3. There are several kinds of possible inconsistencies. It's possible that an entry has some spelling errors in some attributes or accidentally uses values from other entries or its other attributes (i.e. has contradictions to real values of those attributes). How good the validation process can find these contradictions will be evaluated in chapter 6.1. It's also possible that an entry is semantically duplicated to other entry (e.g. the entry uses a synonym name to the name of another entry). The evaluation on the performance of targeting semantic duplications will be talked in chapter 6.2. The

accuracy of feeding solutions to fix inconsistencies is evaluated in chapter 6.3.

6.1 Accuracy of locating contradictions

Based on the degree of how hard to figure semantic contradictions out, the inconsistencies can be divided into two levels. The junior level of contradictions is simple typos or spelling errors which should be comparatively easier to be found out. The senior level is that the attribute itself is a valid one but appears in a wrong entry. The evaluation wants to check the performance of detecting these two levels of inconsistencies for the validation strategy. The F-measure which is a common measure of a test's accuracy is adopted as an indicator of the performance. The F-measure considers both the precision p and the recall r of the test to compute the score.

$$p = \frac{tp}{tp+fp} \quad (8)$$

$$r = \frac{tp}{tp+fn}$$

$$F - measure = \frac{2 * p * r}{p + r}$$

Where the siRNAs involve inconsistency are considered as the positive class, and tp , fp and fn denote the number of true positives, false positives, and false negatives, respectively.

300 randomly chosen siRNAs are used in the evaluation. They were uploaded to CytomicsDB by researchers and are proved to be valid by them. To do the evaluation, 200 siRNAs of the data set is selected and the value of one of the 5 attributes is slightly changed (to a random wrong value by adding, modifying or deleting one base) for each siRNA. To better demonstrate the performance when contradictions happen in different attributes, the 200 siRNAs are divided in five groups, i.e., every 40 siRNAs have errors in a respective attribute. The aim is to see if the validation strategy is able to figure out these intentional errors.

Uploading the 300 records to the test database of CytomicsDB, the measures are listed as following (Fig. 9). The overall F-measure is 0.96 (where the range for F-measure is [0, 1]).

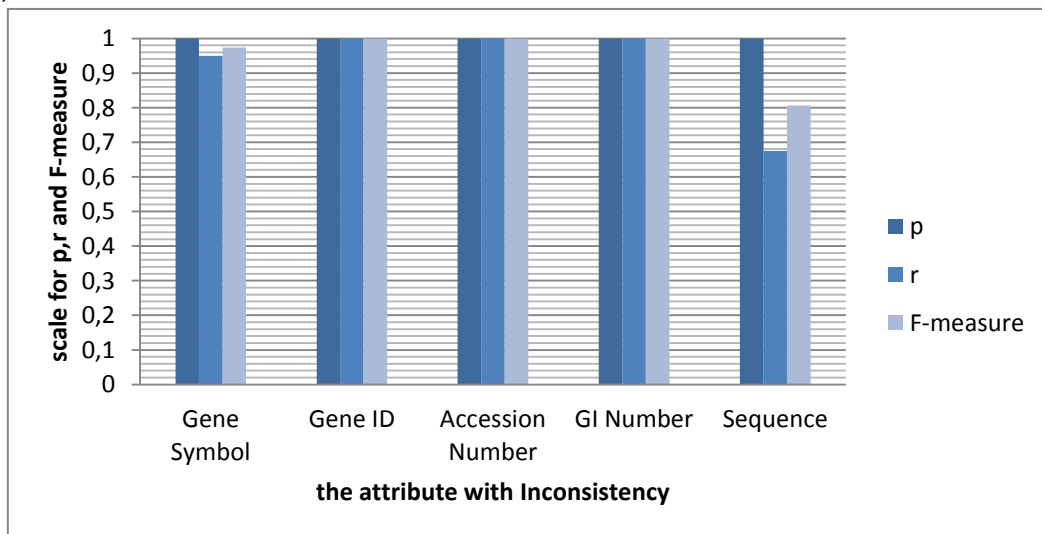


Fig. 9 Evaluation result of the performance of detecting errors

As shown in the result figure, the validation strategy can accurately detect errors in attributes like: Gene ID, Accessing Number and GI Number. There are two false negative cases when the errors are in Gene Symbols. The two are judged as warnings (instead of errors) because the “wrong gene symbols” are accidentally among synonyms. So the validation results still make sense. However, for detecting errors in Sequence, there is still a big space for improvement in the validation strategy.

6.2 Accuracy of locating duplication

100 compound names are adopted in this evaluation. They are from users of CytomicsDB and proved to be inconsistent by them. In this evaluation, synonyms of the 100 compound names are added to the data set as another 100 compounds. So in total, 200 compound names are uploaded to the test database of CytomicsDB. In the 200 compounds, the last 100 compounds should be semantically duplicated to the first 100 ones. Another 100 consistent un-duplicated compound names are attached in the data set after the 200 names to detect possible false positive cases. The aim of the evaluation is to see the performance of the validation strategy detecting these duplex. This evaluation still uses F-measure as the indicator of the performance. The second 100 compound names (which are duplicated) are viewed as positive cases. The measurement result is listed below in Fig. 10.

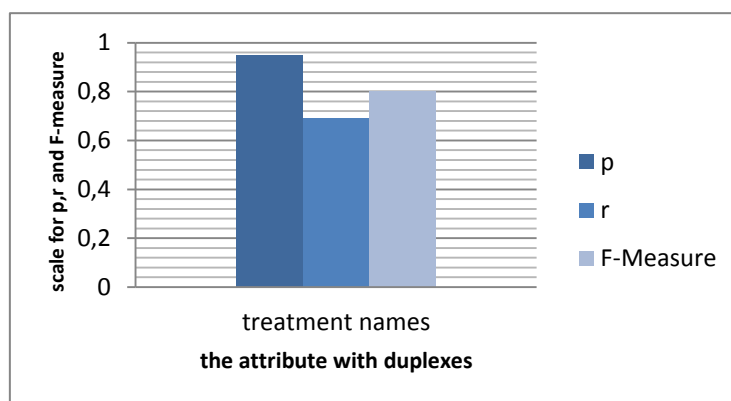


Fig. 10 Evaluation result of the performance of detecting duplexes

Looking into one of the failed case, treatment name “AG-K-27488” which should be a duplex to “MEDRYSONE”, the most possible CID retrieved by the validation strategy is “44308022” but the CID got for “MEDRYSONE” is “247839”. However, the compounds identified by the two CIDs do have the same molecular weight and formula. The structures of them only have slightly differences. This kind of failed cases is quite common in all the false negatives. So it may make sense to have molecular formula as an extra condition (i.e. if CID is same or molecular formula is same then alert the user about the possible duplex) to do the duplex determination.

6.3 Accuracy of giving solutions

In the 185 correctly detected wrong siRNAs from the first evaluation, 162 of them get the original correct siRNA entities as recommended solutions to correct the inconsistencies. The real siRNA entries (i.e. the siRNAs before the intentional modifications) are missing from the recommended solutions for 2 siRNAs with wrong Gene Symbols and 21 siRNAs with wrong sequences. The possible improvement for correcting sequence is going to be discussed in Chapter 7.3.

When the typos in each attribute get more severe (more than one base goes wrong) and more attributes in one siRNA have typos, the recommended solutions from the validation strategy make less sense. When all attribute values in one siRNA are incorrect, the corrections do not reliable any more.

For the 69 true positive treatment names (i.e. truly detected as duplex by the validation strategy) in the second evaluation, the validation strategy correctly points out the treatments they duplicated to for all of them.

7 future works

7.1 Duplex detection for multi-attributes entities

For the siRNAs, now the validation process only checks the value of attribute “Duplex Number” to determine duplexes. However, values in this attribute may not be consistent since they are not validated as well. A similar semantic way of detecting duplex may need to be adopted from the single-attribute entity validation process. That is, if there are two identical candidates stored in the validation details table (where the candidates will be recommended as corrections to users), the user should be warned while they see one of them when they make the decisions. Similar checking and updates as the progress described in chapter 4.1 should be performed for multi-attributes validation as well.

7.2 Multiple external data sources

Now the validation strategy is based on an assumption that the NCBI databases are inner consistent and reliable. However, the assumption is not one hundred percent correct. There can be some overlooked errors while audition or some data loss while backing up or data migration. So only using the NCBI database to retrieve the list of candidates from the list of Accession Numbers is not a quite trustful operation. It could be better to have some other data sources as prove while retrieving candidates. There might be some conflictions during the proving process as well (e.g. the entries from NCBI database and entities from other databases by querying with the same Accession Number might have conflicts). To deal with conflictions in data from multi external data sources, a handy tool might be the Markov Logic network [3]. For example (Table 14), if several siRNA candidates are obtained from different data sources by querying with an Accession Number (NM_003318), the algorithm can be used to construct a candidate which is more trustful than others. In the example, the gene symbol from S1 and S3 are correct but presented in different ways. The gene symbol provided by S2 contains information from S1 and S3. The gene symbol provided by S4 is wrong. The value “TTK” is going to be selected as the trustful value for Gene Symbol attribute if the algorithm runs properly.

Table 14: example candidates from different data sources for NM_003318

Source	Keyword	GI Number	Gene ID	Gene Symbol	Sequence
S1	NM_003318	4507718	7272	TTK	Seq1
S2	NM_003318	34303964	7272	TTK;ESK	Seq2
S3	NM_003318	262399359	7272	ttk	Seq3
S4	NM_003318	675737708	100969041	MPH1	Seq4

The Markov Logic network applies Markov network to a collection of formulas from first-order logic. In the network, the vertices of the network are atomic formulas and the

edges are the logical connectives used to construct the formula. For a set of values $X = (X_1, X_2, \dots, X_n)$, the network is trying to find the most possible one (i.e. find the most possible Gene Symbol value for the attribute from all candidates in the example). There is a weight number for each formula which should be learned by the network during auto-training. The joint distribution represented by the network is:

$$P(X = x) = \frac{1}{Z} \exp(\sum_j w_j f_j(x)) \quad (9)$$

The $f_j(x) \in \{0,1\}$ is each binary formula in the network and w_j is the weight number for each formula. Z is known as the partition function [23]. Here is one example of $f_j(x)$. As a common sense, the value appeared the most frequent in all candidates is more possible to be the accurate value. So for one attribute a , one predicate formula can be defined as $MaxFrequency(a,x) \Rightarrow isAccurate(x)$. The formula can be proved as true or false by each specific test case. The formula can also be defined between data source s and value x as well. Usually the more trustful data source may give the trustworthy value with higher possibility. i.e. $provide(s,x) \wedge isTrustWorthy(s) \Rightarrow isTrustWorthy(x)$. As these formulas use common parameters (like “s” or “a”) and logical connections between them, they can be connected to each other as an undirected graph. The trained Markov Network can predict the true value for each attribute from several candidates. By combining predicted true values together, the final candidate can be generated.

7.3 Sequence correction

For now, it is only possible to check if the siRNA’s sequence existed in its homologous gene’s sequence. If the siRNA’s sequence has several miss-spelled bases which lead to a low alignment score, the validation strategy can only tell the user that there might be some typos in the siRNA’s sequence and give the matched part (which might be shorter than the siRNA’s sequence or might have skips) in its homologous gene’s sequence as a possible correction. To make the recommendation more reliable, a possible solution is involving some siRNA design rules. Then when mismatch happens, the validation strategy may do some guesses on the possible correct siRNA sequence by cross comparing those siRNA sequences generated from design rules with the matched sequence clips. Constructing siRNAs is via finding target areas on a mRNA. One common used rule to find interesting regions (which can be used as siRNA targets) on the mRNA is “Rational siRNA design algorithm” [22]. It identifies eight characteristics associated with siRNA functionality. These characteristics are used to evaluate potential targeted sequences and assign scores to them. Sequences with higher scores will have higher chance of success in RNA interfering. The Table 15 lists the 8 criteria and the methods of score assignment.

Table 15: rational siRNA design criteria (criteria 3: T_m of potential internal hairpin $< 20^\circ\text{C}$)

Criteria	Description	Score	
		Yes	No
1	Moderate to low (30%-52%) GC Content	1 point	
2	At least 3 A/Us at positions 15-19 (sense)	1 point /per A or U	
3	Lack of internal repeats	1 point	
4	A at position 19 (sense)	1 point	
5	A at position 3 (sense)	1 point	
6	U at position 10 (sense)	1 point	
7	No G/C at position 19 (sense)		-1 point

8	No G at position 13 (sense)	-1 point
---	-----------------------------	----------

A sum score of 6 defines the cutoff for selecting siRNAs. All siRNAs scoring higher than 6 are acceptable candidates. With this algorithm, several potential areas on the mRNA sequence can be targeted. Then one of the closest regions to the matched area can be given as a possible correction to the siRNA sequence. More criteria can be applied in the process to narrow down the number of target areas.

7.4 Multi-objective decision

The similarity scores for a candidate of multi-attributes entity can be presented as a vector \vec{x} (for the siRNA's 5 attributes, the dimension of the vector is 5). Accordingly, the weights is a vector \vec{w} which has the same number of dimensions as \vec{x} . Then the sum formula in multi-objective decision is simply described as $U = \vec{w} \cdot \vec{x}$. In the case of validating siRNA, the weight vector is [1,1,1,1,1] by default. However, the weight vector can be tuned by involving the user's previous decisions. So the recommendation result will optimized dynamically based on users selections. Assuming the vector of the user chosen candidate is \vec{c} , then the weight vector can be tuned as:

$$\vec{w}' = (1 - \rho) \times \vec{w} + \rho \times \frac{\vec{c}}{|\vec{c}|} \quad (10)$$

where the ρ is an exponential smoothing parameter which controls the weight vector not to change too violet. While validating the next siRNA entity, the tuned weight vector can be used in multi-objective decision stage.

7.5 siRNA correction

For now, if every attribute in the given siRNA has typos, then the possible correction given by the validation strategy tends to be random. However, for the dataset from users of CytomicsDB, the duplex number can be an auxiliary to give potential solutions when all decision strategies fail.

As described in the beginning of chapter 3, a RNA can have several potential regions which can be siRNA targets. This means that in the master table of CytomicsDB, several siRNAs may correspond to a same homologous gene. This situation has been considered in duplex number. For example, the duplex number "D-004105-01" is actually assembled by two parts. The first part "D-004105" is a registered identifier for the RNA in the lab. The second part "01" means this siRNA corresponds to the first target region. In this case, if there are some typos in attributes (excepts Sequence) which lead to the possible corrections not making sense any more, the validation strategy may try to find a validated siRNA in the master table where the first part of duplex number is the same but the second part is different. Then the Gene Symbol, Gene Id, GI Number and Accession Number from the validated siRNA can be given as a solution to correct the wrong siRNA. This method might be helpful when there are some internal identifier attributes while doing the multi-attributes validation.

Reference

- [1] Larios E, Zhang Y, Cao L, & Verbeek, F. J., CytomicsDB: A Metadata-Based Storage and Retrieval Approach for High-Throughput Screening Experiments[M]//Pattern Recognition in Bioinformatics. Springer International Publishing, 2014: 72-84.
- [2] Boncz P A, Zukowski M, Nes N. MonetDB/X100: Hyper-Pipelining Query Execution[C]//CIDR. 2005, 5: 225-237.
- [3] Yong-Xin Z, Qing-Zhong L, Zhao-Hui P. 2-Stage Data Conflict Resolution Based on Markov Logic Networks[J]. Chinese Journal of Computers, 2012, 1: 010.
- [4] Bleiholder J, Naumann F. Conflict handling strategies in an integrated information system[J]. 2006.
- [5] Bleiholder J, Naumann F. Declarative data fusion—syntax, semantics, and implementation[C] //Advances in Databases and Information Systems. Springer Berlin Heidelberg, 2005: 58-73.
- [6] Levenshtein V I. Binary codes capable of correcting deletions, insertions and reversals[C]//Soviet physics doklady. 1966, 10: 707.
- [7] Kruskal, J. B. (1999). An overview of sequence comparison. In Sanko, D. and Kruskal, J., editors, Time Warps, String edits, and Macromolecules. The Theory and Practice of Sequence Comparison, pages 1{44. CSLI, Stanford, 2ndedition. 1st edition appeared in 1983.
- [8] Chalk A M, Warfinge R E, Georgii-Hemming P, et al. siRNAdb: a database of siRNA sequences[J]. Nucleic acids research, 2005, 33(suppl 1): D131-D134.
- [9] Marler R T, Arora J S. Survey of multi-objective optimization methods for engineering[J]. Structural and multidisciplinary optimization, 2004, 26(6): 369-395.
- [10] Lu J, Zhang G, Ruan D. Multi-objective group decision making: methods, software and applications with fuzzy set techniques[M]. Imperial College Press, 2007.
- [11] E. E. Bolton, Y. Wang, P. A. Thiessen, and S. H. Bryant. Chapter 12 pubchem: Integrated platform of small molecules and biological activities. volume 4 of Annual Reports in Computational Chemistry, pages 217 – 241. Elsevier, 2008.
- [12] I. Mizrahi. Chapter 1 genbank: The nucleotide sequence database. In J. McEntyre and J. Ostell, editors, The NCBI Handbook [Internet]. Bethesda (MD): National Center for Biotechnology Information (US), 2002.
- [13] Hannon G J. RNA interference[J]. Nature, 2002, 418(6894): 244-251.
- [14] Bruford E A, Lush M J, Wright M W, et al. The HGNC Database in 2008: a resource for the human genome[J]. Nucleic acids research, 2008, 36(suppl 1): D445-D448.
- [15] Johnson M, Zaretskaya I, Raytselis Y, et al. NCBI BLAST: a better web interface[J]. Nucleic acids research, 2008, 36(suppl 2): W5-W9.
- [16] Sayers E. The E-utilities in-depth: parameters, syntax and more[J]. 2014.
- [17] Benson DA, Cavanaugh M, Clark K, et al. GenBank[J]. Nucleic acids research, 2012: gks1195.
- [18] Maglott D R, Katz K S, Sicotte H, et al. NCBI's LocusLink and RefSeq[J]. Nucleic acids research, 2000, 28(1): 126-128.
- [19] Sayers E. E-utilities quick start[J]. 2013.
- [20] E. Larios, Y. Zhang, K. Yan, Z. Di, S. LeD'ev'edec, F. Groffen, and F. Verbeek. Automation in cytomics: A modern rdbms based platform for image analysis and management in highthroughput screening experiments. In Proceedings of 1st Int. Conf. on Health Information Science, volume 7231, pages 76–87, 2012.
- [21] Goncalves A. Java Persistence API[M]//Beginning Java EE 7. Apress, 2013: 103-124.
- [22] Reynolds A, Leake D, Boese Q, Scaringe S, Marshall WS, Khvorova A. Rational siRNA design for RNA interference. Nat Biotechnol. 2004 Mar;22(3):326-30.
- [23] Richardson M, Domingos P. Markov logic networks[J]. Machine learning, 2006, 62(1-2): 107-136.

Appendix

Link.1: the RNA sequence of NM_003318.1 (identifier: accession number)

http://www.ncbi.nlm.nih.gov/nuccore/NM_003318.1

Link.2: the RNA sequence of NM_003318.2 (identifier: accession number)

http://www.ncbi.nlm.nih.gov/nuccore/NM_003318.2

Link.3: the RNA sequence of NM_003318.3 (identifier: accession number)

http://www.ncbi.nlm.nih.gov/nuccore/NM_003318.3

Link.4: the RNA sequence of NM_003318.4 (identifier: accession number)

http://www.ncbi.nlm.nih.gov/nuccore/NM_003318.4

Link.5: the RNA sequence of XM_008969441.1 (identifier: accession number)

http://www.ncbi.nlm.nih.gov/nuccore/XM_008969441.1

Link.6: the RNA sequence of NM_001166691.1 (identifier: accession number)

http://www.ncbi.nlm.nih.gov/nuccore/NM_001166691.1

