User-defined view automation of genomic databases

Andrew J. Robinson, Wenny J Rahayu
Dept. of Computer Science
Latrobe University, Bundoora
Victoria 3083, Australia
aj5robinson@cs.latrobe.edu.au

Susanna Herd
Dept. of Biochemistry
Latrobe University, Bundoora
Victoria 3083, Australia
S.Herd@latrobe.edu.au

Tharam Dillon
Faculty of IT
University of Technology Sydney
NSW, Australia
tharam@it.uts.edu.au

ABSTRACT
This paper presents a solution to the problem of creating a subset database from the public genome databases, also known as a database view. While the techniques to generate views are well established already in the database system there are still some problems found where applying this technique in the genome database environment. The main problems that exist in the current methods of view creation are missing relevant results, returning irrelevant results and view creation processes are generally very time consuming for the user. The solution presented within provides an automated approach aimed at reducing the time needed to create a view, which is usually done by hand. The solution improves the searching method needed for view creation by the addition of two extra phases; the first, expanding the keyword search so that it captures all relevant results and second, a filtering phase to remove all the extra irrelevant results. The whole process is done in the background so that the user isn’t required to spend much time fixing the results of inadequate search tools.

Categories and Subject Descriptors
H.2 [Database Management]: Heterogeneous Database - Data Translation and Program Translation.

General Terms
Algorithms, Design.

Keywords
Genome database, ontology, database views.

1. INTRODUCTION
With the introduction of mass genome sequencing machines and projects comes a large volume of sequence and annotation data. Much of the work that scientists in the area of bioinformatics involves the use of subsets of the data produced from these sequencing projects also known as database views.

The main reasons that views have been created for genomic databases are (1) They can reduce the size of large databases and thus make them more directed to the research at hand, (2) the user can customize their view to include extra annotation and remove non-relevant parts of annotation and (3) a view can be an integration of several external databases so that it can be more complete.

Currently the process of creating views consists of running a query (typically a keyword search) on an external database and the results form the basis of the view. The view usually needs to be refined due to the inaccuracy of the typical keyword search. Other results are added when updates occur on the external databases or from research in the biology lab. This process has many problems such as retrieving irrelevant results, missing relevant results and thus leading to much manual filtering being required. Current work in this area does not attempt to cover the integration mentioned above for using views but work has been done which solves the issues relating to compatibility and access to multiple databases from a single point (or program) [1, 2, 3]. However having said that, there remains a need for automated creation of views of this combined data.

The solution presented here uses our previous work in [1] to combine and connect the external databases so that the more complete view can be created. To expand the search and also again filter the results it uses a word ontology called Wordnet [4] and other domain specific Ontology’s such as Gene Ontology [5].

Section 2 of this paper provides an overview of related work in the area of view creation and the issues that need to be solved. Section 3 defines an algorithm to solve the issues. Section 4 gives a working example / proof of concept which is followed by a conclusion in Section 5.

2. BACKGROUND
Currently we can categorize three main techniques where views of public genome databases can be created. The three types are referred as Pre-constructed, Manual view creation and Semi-Automatic view creation in this paper and are depicted in figure 1.

The 3 basic types of view creation are:
1. Pre-constructed: They tend to be very limited in the views that are made available by the database provider. They are usually created by source species (e.g. mouse, canola etc.) and subdivided by sequence type (e.g. DNA, BAC, etc.). The Swiss Prot database provides views of this type [7].
2. Manual view creation: Consists of searching each external database separately and combining by hand to remove redundancy (and some times irrelevant data from the view). As
the name suggests this method requires much manual work and thus no researchers actually publish works of this type. However this does not mean that biologist do not use this method to create views. From personal experience it would be quite likely that the majority of views have been created using this method.

3. Semi-Automatic: Same as type (2) except that the searching, combining and redundancy parts are done automatically from a single search given by the user. The majority of Genomic database integration attempts fall into this category [1, 2, 3].

![Diagram showing manual and semi-automatic view creation]

Figure 1: Current methods of view creation

The ‘Pre-Constructed’ view creation is where the public database provider groups their sequences into several groups. These are usually very general group such as species and sub groups of the types of sequences (e.g. DNA, Genomic etc.). The process of creating the view is very simple; just download the required view from the internet. In the figure there are multiple available views on the left and the user simply selects one or more to make their view from; depicted on the right by the larger box. Generally this method of view creation returns large views with many irrelevant results. Another problem with this method is it is not customizable due to the static nature to the views.

The ‘Manual view creation’ method consists of the user executing keyword or other searches on multiple public databases and combining these manually. The combining phase generally is very tedious because the user has to remove redundancy in many cases. And when added to the manual filtering required to remove irrelevant results, the task becomes a nightmare.

The final method of view creation is ‘Semi-Automatic view creation’. This method is very similar to that of the ‘manual view creation’ except that in this case the redundancy removal has now been automated. This method has been called ‘semi-automatic’ because the user still has to create a search and make sure that it returns what they are looking for and possibly run another search.

As can be seen in Figure 1 there are several problems with the current methods for creating a view.

The problems are as follows:

1. They miss many relevant results (false negatives). An example this problem is when a sequence is annotated with a keyword synonym. So when the keyword search checks this result it misses it because it does not contain the actual keyword.

2. They return irrelevant results (false positives). An example of this is often encountered in view creations is where a view of sequences contain a certain protein. Creating this view would require a keyword search for the protein’s name. This would return all sequences which have documentation saying this sequence contains the protein. However it would also include sequences which annotation says it is similar to that protein (a common practice in the genomic databases).

3. Large database so manual refinement is very inefficient. An Example is the human DNA sequence database being in excess of 2 GiB in size.

3. PROPOSED USER-DEFINED VIEW ALGORITHM

Much current work only attempts to solve the issues with integration of the external database and not actually improve the search capabilities [10, 11]. Most of the more advanced work seems to only produce a tool kit of objects and converters that make it ‘easier’ for the user to create their own new modules. While these tool kits are sometimes easy to use they are not developed enough for the user to create views.

Many results are missed with current view creation techniques because the searches involved are not broad enough to capture all results. Since much of the sequences in the genomic database have detailed annotation attached we have chosen to use an approach that makes use of this. The approach uses the Wordnet Ontology along with Gene Ontology to expand the search for relevant sequences for the view. “An ontology defines a common vocabulary for researchers who need to share information in a domain” [8]. In this case the domain is Annotation of sequences in Genomic Databases using the English language. The main use of the Ontology’s is to obtain synonyms and parent-child relationships between the keywords used for searches and the annotation of the sequences.

The second problem identified above can typically be solved by an additional filter being connected to the process of view creation. As with the first problem we have chosen to make use of the abundant annotation provided along with the results to filter the irrelevant results from the view. Our solution uses a language analysis tool to divide the results into several groups based on the context in which the keyword is used in the annotation. Following this the user is given the option to select which groups they wish to include in their view.

The third problem is addressed inherently while solving the problems one and two. By making the search much more accurate, the need for repeating a view creation is reduced dramatically as well; thus removing the need to address the problem directly.

The algorithm that we have constructed consists of three phases. Each of these will be described in more detail in the following sections.

The first and third phases of the algorithm solve the first and second problems respectively. The second phase is simply to automate the search and integrate the results from multiple public databases.
3.1 Phase 1: Expanded requirement keywords
To overcome the first problem of view creation the initial search needs to be expanded. The main need for this is to make sure that all the relevant results are returned. This aim is completed by getting synonyms from an ontology such as Wordnet [4]. Doing this allows the search to be simple but return the result of a complex search. To make sure that the synonyms are relevant each is presented to the user for confirmation. This change is depicted by phase one in figure 2. While the aim of this phase is to return all relevant results it does however capture more irrelevant ones too. The irrelevant results are then handled by phase three. Step 1 can be simply a text box in which the user types a list of keywords or a complex interface which assists the user in coming to the required list of keywords. Step 2 searches the ontology’s for synonyms of the keywords used. And step 3 gives the user a list of synonyms from which to choose. This final step could be skipped if some extra intelligence is added to allow the computer to determine what the user really wants.

3.2 Phase 2: View creation
Phase 2 in figure 2 below is similar to that of a standard keyword search over multiple public databases. The only difference is that the initial keywords have been expanded by phase one. Step 1 of this phase simply provides a list to the user of all available databases. Step 2 converts the list of keywords into a query that can be executed on the external databases. The query constructed is one that returns results if it has one or more of the keywords in the annotation. Step 3 executes the query on the external database and converts all the results to a common format for combining. Finally Step 4 combines the results from all the external databases in a non-redundant way, i.e. compare the IDs and merge results that have the same IDs or other identifiers.

3.3 Phase 3: Optimize view
To remove the false positives a filtering phase has been added. This phase searches the fields from the results for sentences containing the keywords. Upon finding one or more keyword(s), the sentence is analyzed. The analysis extracts the subject and predicate from the sentence. Predicates are grouped into several groups; these are ‘negative or not like the subject’, ‘similar or like the subject’ and ‘positive or equal to the subject’. At the beginning of this phase the user is asked which of these groups they wish to have in their view. The subject and predicate can be found by analyzing the words that make up the sentence [5] or computationally with [9]. The process involves working out the type of each word (e.g. Noun, verb etc). Isolating the verb of the sentence shows the subject (before the verb) and the predicate (after and including the verb). In this algorithm we have used Wordnet and another domain specific ontology called Gene Ontology (GO) [6] to find the types of the words.

In Step 1 the system asks the user for the context(s) that the user wants in their view; this is done for each keyword in the list although the option is give to use the same context of all keywords. Step 2 uses the method described before to analyze the sentences to determine which of the 3 context groups to which it relates. Finally Step 3 looks at all the results and places them into the required groups as per the meanings of the sentences. Discrepancies are resolved by placing the result in the highest context found in the annotation. The ordering of context groups is by their degree of similarity to the one of Keywords; the most similar result coming first and ending with one meaning not similar. E.g. the first group might be one meaning an exact match and the last group being no similarity at all.

3.4 Solution Overview
Figure 2 above shows the three main phases of the ‘user-defined view creation algorithm’. It also shows each of the main steps that are needed to complete each phase as described in the previous sections.

Figure 3 above shows the architecture that was presented in previous works [1] in single lines, with double lines highlighting our work. As depicted in the figure the solution presented here is implemented in a Task layer called view creation with an added Service of the same name to advertise the new functionality to the User interface program.

3.5 Detailed description of solution
Figure 4 presents a more detailed explanation of our solution. It is provided in a Pseudo Code style of text.

Phase 1:
Read Keywords from user
For each keyword
Call function “Get Synonyms” (keywords as parameters)
If keyword is made up from multiple words
Call function “Get Synonyms” (multiple keywords as parameters)
End if
Ask user which keywords they want
End for each
Phase 2:
Show list of connected databases to user
Get databases to search from user
Construct the query from each of the keywords
‘OR’ed’ together
For each database
Run query on database
Get results from query
For each result
Add a lookup entry for the result’s ID
If result is already in lookup table
Combine new annotation with current result
End if
End for each result
End for each database
Phase 3:
Show list of available contexts to user
Get which contexts they want
For each result
For each field in the annotation that contain
English words
Find keywords
If keywords found
Call Function “Analyze sentence” (sentence as
parameter)
End if
End for each field
Calculate highest context group for this result
belongs to. (Ascending order, not like, like, exactly)
Place result in a list for this context.
End for each result
Return the lists containing the required contexts.

Function “Get Synonyms” (keywords as parameters)
For each ontology
For each keyword
Find the keyword in the ontology
If keyword exists
Collect all other keywords that have
“synonym” link to this keyword
End if
End for each keyword
End for each ontology

Function “Analyze sentence” (sentence as
parameter)
If sentence contains a keyword
Run external sentence meaning extractor
For Highest to Lowest keyword meaning group
If sentence means sequence is in this group
Return this group
End If
End For
Else
Return ‘not relevant’
End if

4. DEVELOPMENT OF PROTOTYPE
A common scenario of using this system will now be presented to show how the user interacts with the system. For this scenario the user wishes to find ‘14-3-3 protein’. After typing the ‘view keywords’ the system expands them to ‘14-3-3 (protein | super molecule | macro molecule)’. The user decides that ‘14-3-3 protein’ and ‘14-3-3 super molecule’ are useful.

For phase 2 the user decides they want the view to be created from ‘swiss prot’ and ‘trembl’ [7]. Executing this search on ‘swiss prot’ and ‘trembl’ returned 222 results.

For phase 3 the user wants all results that match the keywords exactly and not ‘similar’ or ‘like’. After processing the system found 147 results that were in the ‘exactly’ group which are kept for the view and 75 in the ‘similar/like’ group which are discarded.

Figure 4: Solution Pseudo code
Figure 5: Pictorial view of the ‘user-defined view creation algorithm’.

Figure 5 above follows the same conventions as Figure 1 and can be compared to discover the differences in the algorithms. The key things to note from figures 2 and 3 are that added expanding and optimizing phases; phases 1 and 3 respectively.
The user is then shown a list of databases that they can create their view from. Figure 9 shows the list which the user can select from. This list is based on which databases have been connected to the mediator in the architecture.

The user is then given a list of different meaning groups to create their view from. The groups were explained earlier. Figure 10 shows the selection list which the user can use. Finally Figure 11 shows how the results are shown to the user. Alternatively this information could be saved into a file on the client-side computer or within the mediation layer as described in the previous paper.

5. CONCLUSION

In the earlier sections a solution was presented to the problem of creating a view from the public genome databases. The main problems that needed to be addressed were (1) missed relevant results (2) returned irrelevant results and (3) large databases requiring minimal search repeats. The first problem was addressed by the addition of phase 1, an expanded keyword search so that it returns more results that capture all relevant results. This brought about more irrelevant results being returned but this is handled with the other irrelevant results by adding the filtering phase 3. The filtering phase uses a sentence meaning extractor which was developed using Wordnet and Gene Ontologies. The system grouped the results according to what the annotation meant and allowed the user to select which meaning they wish to include in their view. From here further work can be completed in order to optimize the meaning extractor as it took a fair time to run on datasets given. Also the meaning extractor could analyze the all sentences, not just the ones containing the keywords. This would allow inferred knowledge to be captured.

6. REFERENCES