Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Date: 4 Dec. 2018
CSE 549 Paper

Pathway Analysis

KEGG Pathway Navigator, an open-source web app for extracting pathway genes

Richard Sicoli [1], Richard Yao [1] and Bladimil Nunez [1]

[1] Department of Computer Science, Stony Brook University, Stony Brook, NY 11794, USA

Abstract

Motivation: Performing pathway analysis is a vital part of many bioinformatics pipelines. There are not many tools that exist to make this process easier. To our knowledge, the only existing tool, KPGminer, is not open-source and only available as a Windows executable. We propose a modern, open-source, and cross-platform system, KEGG Pathway Navigator, for streamlining the extraction of pathway constituent genes from the KEGG database.
Results: We find our tool successfully manages to streamline the retrieval of pathway genes from various organisms by making use of the KEGG REST APIs. We also successfully implement all of the functionality of the existing KPGminer tool plus additional features and optimizations. We find our system outperforms the existing KPGminer both in pathway retrieval time and in overall functionality.
Availability: Our system is completely open-source and currently hosted as a web application for other academics to use. Our system is built as a web application with a Python and Flask backend and the use of HTML/CSS/JavaScript for the interface.
Contact: rpsicoli@cs.stonybrook.edu
Supplementary information: The web app is hosted at: https://richsbcs.pythonanywhere.com
The source code is available at: https://github.com/richard-sbcs/KEGG-Pathway-Navigator

1 Introduction

Pathway analysis is an essential part of bioinformatics and computational systems biology. Various biological pathway and gene databases exist, such as KEGG [5], Reactome [4], GO [1], and Wikipathways [6]. The scope of our work focuses of the KEGG Pathway database.

KEGG [5] (Kyoto Encyclopedia of Genes and Genomes) is one of the popular databases used in bioinformatics and computational biology systems. The KEGG database stores data relating to biological pathways, genomes, diseases, drugs, and chemical substances. The KEGG database exposes a number of useful RESTful APIs to make developing and integrating tools feasible. Our work focuses specifically on the extraction of biological pathways of various organisms in the KEGG pathway database.

Biological pathways consist of a set of related genes and proteins that perform biological functions such as signal transmission, gene expression regulation and metabolism. Pathway databases like KEGG store the different types of pathways (e.g. signaling, diseases, genetics, etc.) for a vast collection of different organisms.

Many bioinformatics and computational biology pipelines depend on annotated pathways for pathway analysis. This may include performing enrichment tests on new gene sets compared with the annotated pathways from a database, or analyzing gene set overlap. As discussed in [2], previous work demonstrates the use of enrichment tests for cancer module identification [3]. Evidently, access to annotated pathways in a streamlined manner is critical for pathway analysis.

To our knowledge, there is currently no open-source, cross-platform solution that achieves streamlined retrieval of KEGG pathway genes. The existing tool, KPGminer [2], is only available as a Windows executable. Not only is it not cross-platform, we found numerous software bugs and usability issues with the tool. For example, we found issues with the tool's parsing of organism names and pathway genes. We also found it lacks essential functionality such as search.

For these reasons, we propose a novel open-source and cross-platform system, KEGG Pathway Navigator (KPN). KPN uses modern web application technologies and design principles to build a truly streamlined tool for the extraction of pathway genes. We believe the tool can be a useful part of the bioinformatics pipelines used by researchers and academics.

We implement various types of additional functionality into our system to significantly improve the user experience. This includes features such as real-time search, pathway descriptions, configurable download options, various file types for exporting, and modern UI design principles.

We also develop a number of optimizations to improve our system's performance over the existing work. We exploit KEGG batching optimizations to reduce the time for full pathway retrieval of an organism by roughly a half. We also exploit shared caching to reduce the time of previous requests to near instantaneous.

2 Approach

Our initial approach was divided into three main goals. First we wanted to determine the technology we were going to use for the tool. Secondly, we wanted to determine the functionality that the tool would perform. Finally, we developed mockups of the interface to plan how the user interface and interactions would work.

2.1 Technology

Our approach to the problem was to first identify a technology that would work cross-platform, so that any user could run our tool. We decided on a web application for a few reasons:

- **Inherently Cross-Platform**: Web applications run in a browser, so they are inherently cross platform. This was the ideal solution, as we didn't need to handle any native system dependencies.
- **Shared Caching**: Given that our web application can handle requests from a variety of users, we can actively cache any requests to the KEGG database. This allows us to deliver near instantaneous requests to users who issue a request for data that is fully cached or near-fully cached. This also has the benefit of reducing the overall load on the KEGG database server itself, as users who make duplicate or partial-duplicate requests can receive the cached results.
- **Truly Streamlined**: Web apps are extremely easy to access. They require no setup or installation by the user. They integrate well with automated tools as well.

2.2 Functionality

The next part of our approach was to determine what kinds of functionality to implement. We knew we had to re-implement at a minimum the features of the KPGminer tool. Those features are defined below.

- **Organism List**: When the tool launches, it presents a list of all organisms in the pathway database. The user first starts by selecting the desired organism before moving on to pathway extraction.
- **Pathway Retrieval**: Once the user has selected an organism, the tool loads all pathways for the given organism from the pathway database.
- **Pathway Selection Control**: The user has fine-grain control over what pathways they are interested in. The user can select individual pathways or choose to select them all in batch mode [2]. The user can also choose to remove individual selections or clear all selections.
- **Pathway Information**: The tool will display the information relating to each selected pathway in a table. This includes the pathway name, description, and most importantly the list of genes.
- **Downloading to File**: Once the given pathways are selected, the user can export the results to a standard tab-delimited .gmt file for further processing.

We also came up with a list of additional functionality and improvements that we would want to integrate. These include the following:

- **Dynamic Search**: Our approach to searching was to allow users to find what they are looking for as quickly as possible. We allow searching through the thousands of organisms from the KEGG database as well as the pathways for each organism. Searching is done in real-time, so the search results update as the user types. Searching is also smart, so it will not only filter by the entity name but also by the organism prefix code or pathway ID.
- **Pathway Descriptions**: Along with the pathway name and genes, we also store the description of the pathway as well. We noticed that KPGminer had a blank column for this, so we decided to fill it with the descriptions from the KEGG database.
- **Configurable Download Options**: Our approach for downloading was to include the original functionality of KPGminer, but also to include a few additional customizable options which we think researchers might find useful. This includes an option to include descriptions in the download file. This is useful if you are only interested in the pathway genes and don't want to waste space storing the descriptions. We also have two options for how to download the results to a file. First is the original tab delimited .gmt file. This is the same format that KPGminer uses and is used by MSigDB [7]. Second is the alternative option to download the pathway data to a .csv file.
- **Full Screen Results and Highlighting**: We also wanted to improve the way results are displayed to the user. We made sure to have the results consume the entire window and also highlight the pathways to make reading them easier. So even if a user is not interested in downloading to a file, they can still easily view and navigate the pathway results.

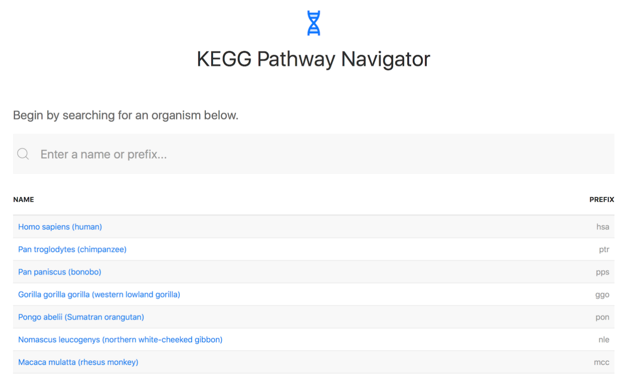
2.3 Interface

Our approach to the interface was to first determine the types of pages that the user would interact with. We came up with a design of three main pages. These pages are shown in figure 1 and are described below.

- **Organism Selection Page**: The organism selection page is the very first page the user interacts with when using the web application. The user is able to search the list dynamically for organisms by name or pathway prefix and select the desired organism to view the pathways for.
- **Pathway Selection Page**: The pathway selection page includes two tables. One which includes all of the pathways for the given organism, and the other table represents the pathways which the user has selected to retrieve. This functionality is similar to how the existing tool, KPGminer, handles the pathway selection. It allows the user to first find the pathways they are interested in, and then submit a request to retrieve the pathways. The user can also choose to select multiple or all pathways and remove their selections as well.
- **Results Page**: The results page includes the pathway results for the given organism based on the biological pathways that the user has selected. The results show the pathway name, the pathway description and the pathway genes for each selection. The results page also includes the options for downloading to file, as discussed previously.

By dividing the user interactions into three separate pages, it allowed us to display more information without overwhelming the user. It also makes the process more intuitive for the user, as the user can focus on each task sequentially. We also made sure to adopt modern UI principles, such as creating responsive layouts, proper typography, and appropriate spacing for improved readability. The end result is an interface that correctly fits to the given page size and is easy for the user to follow.

We also knew that we wanted to design the interface with real-time search in mind. So we made sure to properly plan a layout that could allow the user to quickly search for organisms and pathways, and have the results update in the table dynamically. We also took steps to develop the interface with the system status in mind. For example, we use loading indicators and warning messages to indicate the current state of the request.

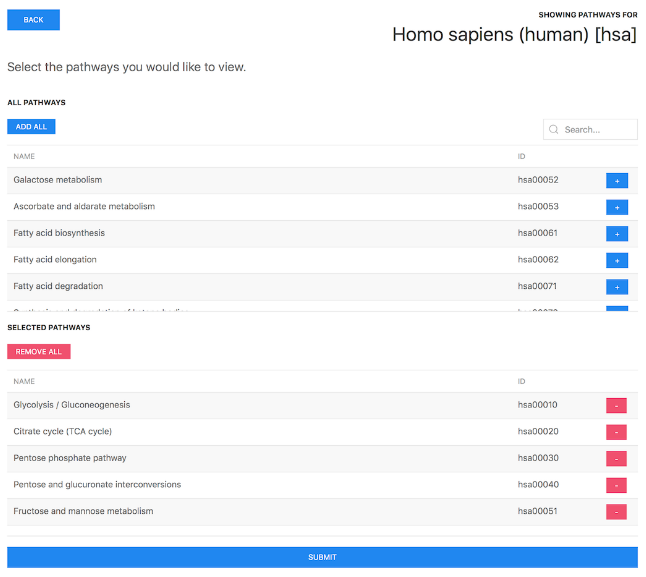

KEGG Pathway Navigator

Begin by searching for an organism below.

Enter a name or prefix...

NAME	PREFIX
Homo sapiens (human)	hsa
Pan troglodytes (chimpanzee)	ptr
Pan paniscus (bonobo)	pps
Gorilla gorilla gorilla (western lowland gorilla)	ggo
Pongo abelii (Sumatran orangutan)	pon
Nomascus leucogenys (northern white-cheeked gibbon)	nle
Macaca mulatta (rhesus monkey)	mcc

Fig. 1. Organism Selection Page



Homo sapiens (human) [hsa]

Select the pathways you would like to view.

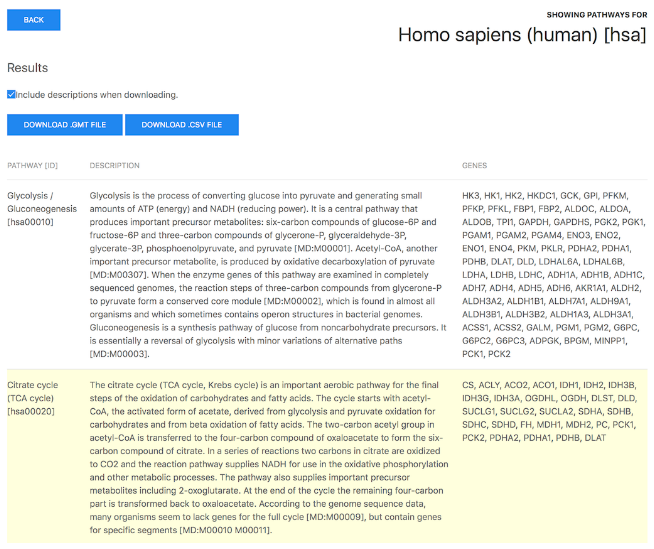
ALL PATHWAYS

NAME	ID
Galactose metabolism	hsa00052
Ascorbate and aldarate metabolism	hsa00053
Fatty acid biosynthesis	hsa00061
Fatty acid elongation	hsa00062
Fatty acid degradation	hsa00071

SELECTED PATHWAYS

NAME	ID
Glycolysis / Gluconeogenesis	hsa00010
Citrate cycle (TCA cycle)	hsa00020
Pentose phosphate pathway	hsa00030
Pentose and glucuronate interconversions	hsa00040
Fructose and mannose metabolism	hsa00051

Fig. 2. Pathway Selection Page



Homo sapiens (human) [hsa]

Results

[✓] Include descriptions when downloading.

PATHWAY [ID]	DESCRIPTION	GENES
Glycolysis / Gluconeogenesis [hsa00010]	Glycolysis is the process of converting glucose into pyruvate and generating small amounts of ATP (energy) and NADH (reducing power). It is a central pathway that produces important precursor metabolites: six-carbon compounds of glucose-6P and fructose-6P and three-carbon compounds of glycerone-P, glyceraldehyde-3P, glycerate-3P, phosphoenolpyruvate, and pyruvate [MD:M00001]. Acetyl-CoA, another important precursor metabolite, is produced by oxidative decarboxylation of pyruvate [MD:M00307]. When the enzyme genes of this pathway are examined in completely sequenced genomes, the reaction steps of three-carbon compounds from glycerone-P to pyruvate form a conserved core module [MD:M00002], which is found in almost all organisms and which sometimes contains operon structures in bacterial genomes. Gluconeogenesis is a synthesis pathway of glucose from noncarbohydrate precursors. It is essentially a reversal of glycolysis with minor variations of alternative paths [MD:M00003].	HK3, HK1, HK2, HKDC1, GCK, GPI, PFKM, PFKP, PFKL, FBP1, FBP2, ALDOC, ALDOA, ALDOB, TPI1, GAPDH, GAPDHS, PGK2, PGK1, PGAM1, PGAM2, PGAM4, ENO3, ENO2, ENO1, ENO4, PKM, PKLR, PDHA2, PDHA1, PDHB, DLAT, DLD, LDHAL6A, LDHAL6B, LDHA, LDHB, LDHC, ADH1A, ADH1B, ADH1C, ADH7, ADH4, ADH5, ADH6, AKR1A1, ALDH2, ALDH3A2, ALDH1B1, ALDH7A1, ALDH9A1, ALDH3B1, ALDH3B2, ALDH1A3, ALDH3A1, ACSS1, ACSS2, GALM, PGM1, PGM2, G6PC, G6PC2, G6PC3, ADPGK, BPGM, MINPP1, PCK1, PCK2
Citrate cycle (TCA cycle) [hsa00020]	The citrate cycle (TCA cycle, Krebs cycle) is an important aerobic pathway for the final steps of the oxidation of carbohydrates and fatty acids. The cycle starts with acetyl-CoA, the activated form of acetate, derived from glycolysis and pyruvate oxidation for carbohydrates and from beta oxidation of fatty acids. The two-carbon acetyl group in acetyl-CoA is transferred to the four-carbon compound of oxaloacetate to form the six-carbon compound of citrate. In a series of reactions two carbons in citrate are oxidized to CO2 and the reaction pathway supplies NADH for use in the oxidative phosphorylation and other metabolic processes. The pathway also supplies important precursor metabolites including 2-oxoglutarate. At the end of the cycle the remaining four-carbon part is transformed back to oxaloacetate. According to the genome sequence data, many organisms seem to lack genes for the full cycle [MD:M00009], but contain genes for specific segments [MD:M00010 M00011].	CS, ACLY, ACO2, ACO1, IDH1, IDH2, IDH3B, IDH3G, IDH3A, OGDHL, OGDH, DLST, DLD, SUCLG1, SUCLG2, SUCLA2, SDHA, SDHB, SDHC, SDHD, FH, MDH1, MDH2, PC, PCK1, PCK2, PDHA2, PDHA1, PDHB, DLAT

Fig. 3. Results Page

3 Methods

The implementation of our web application is divided into two main parts. The backend system and the frontend browser technologies. We will now discuss the implementation methodologies for each part in detail.

3.1 Backend Implementation

3.1.1 Setup

For our backend, we are using python3 with the Flask web development framework. We also use SQLite as our data store. We also currently host our system on pythonanywhere.

3.1.2 Static Routes

We set up static routes for the webpages and JavaScript source files. These pages contain the interfaces discussed in section 2.3. Note that these pages do not include any information from the KEGG database. The data is asynchronously fetched from our KEGG Pathway Navigator API which we use to serve the results to the user. This process is discussed in the next section.

3.1.3 KEGG Pathway Navigator API

We set up a light API for handling requests to our server and preprocessing the data from the KEGG database. We begin by first fetching the data using the KEGG Pathway RESTful API using various HTTP requests. We then parse the data to extract the relevant information, such as organism names, organism prefixes, pathway names, pathway IDs, pathway descriptions and pathway genes.

We cache these results temporarily in our key-value store for future requests. We make sure to store request data individually, so that partial caching can occur if some of the request overlaps with previously requested data.

This data is then returned in a JSON format to the client where it is further processed and handled by the frontend.

3.1.4 KEGG Pathway Database API

As discussed previously, our system makes heavy use of the KEGG Pathway Database REST API. We use HTTP requests to documented endpoints on the KEGG database server to extract relevant pathway information. We only need to contact the KEGG online database for requests that have not been cached.

3.1.5 Optimizations

We take many steps to optimize the performance of our backend. The longest delay in time is the time it takes to send and receive a request from the online KEGG database server. To handle this, we use caching to store previously processed requests. This allows us to serve large pathway requests near instantaneously.

The other major optimization we take is in how we handle batched requests. We exploit the KEGG database <dbentries> identifier to serve multiple pathway retrievals in just one request. We found the hard limit of the API to be set at 10. This allows us to batch multiple pathway retrievals into maximum groups of 10, allowing us to handle large non-cached pathway requests much faster. This is particularly useful when needing to request all of the pathways for a given organism. We discuss the performance of this technique in the results section 4.1.3.

3.2 Frontend Implementation

3.2.1 Setup

For our frontend, we are using standard HTML/JavaScript/CSS. We also use UIKit 3 for our modern interface design, and Clusterize for rendering large table data (e.g. thousands of organism entries) efficiently in the browser.

3.2.2 Presentation

Our presentation layer is divided into two content categories: static and dynamic content.

- **Static Content**: This includes the HTML elements and JavaScript code that are served with the initial page request. This includes basic elements such as the page titles, page layouts, loading indicators, etc. We use three separate static pages for the user interface (as discussed in section 2.3). This content loads first before any further processing on the front end.
- **Dynamic Content**: This includes the content that we generate dynamically using JavaScript code. There are two scenarios where this is necessary.

  Firstly, the JSON data we receive from our server is converted to and rendered as HTML elements dynamically. We use the Clusterize library for rendering large table data efficiently as fixed sized tables. The one exception to this is the results page, where we display all of the results on the entire page at once to make navigating the results easier as discussed in the "Full Screen" functionality in section 2.2.

  Secondly, the real-time search content is also generated dynamically on the client side by filtering and rendering HTML elements using the given search query.

3.2.3 Asynchronous Requests

Pathway and organism information from our server is not automatically included in the web pages we serve. Instead, a static initial page is served, followed by an asynchronous fetch request for the data. This is done for two reasons:

- **Data Size**: Large amounts of data is often returned by our backend server. This could be a large list of organism names, or a large number of pathways for an organism. Because this data can be large it can take time to process and transfer, so we first start by immediately loading the page with static content first, then use a loading indicator to indicate the asynchronous data transfer that occurs in the background.
- **Request Latency**: If requested data is not cached on our server, we need to send a response to the KEGG pathway database. These requests take time, especially if they require multiple batched requests to the KEGG database. Rather than having the entire webpage stall for the request, we use asynchronous fetching to load the data in the background.

3.2.4 Optimizations

As discussed previously, we use asynchronous requests and the Cluzterize library to improve browser performance. We also, however, make additional optimizations in the browser. For instance, we store local pathway data at the client once it is retrieved to prevent reloading data when the user switches between the results page and the pathway selection page. This allows the user to quickly view the results for different pathways of the same organism without needing to encounter a reload.

4 Results

We have results for how our system compares with existing work, and we also have results on request limits using concurrent connections to the online KEGG Pathway database.

4.1 Comparison with KPGminer

First, we compare our system, KEGG Pathway Navigator to the existing work, KPGminer. This is important as the primary goal was to develop an open-source and cross-platform solution that at a minimum provided the functionality of KPGminer. We compare our system across three categories: design, functionality, and performance. We used Windows 10 when running KPGminer.

4.1.1 Design

While comparing designs can often be subjective, there are some clear advantages of our KEGG Pathway Navigator over KPGminer. For starters, we follow modern UI design principles by paying close attention to standard interface guidelines such as responsive layouts, formatting content properly, adjusting typography for certain emphasis, and using appropriate spacing to improve readability.

We have also found some design issues with KPGminer. For starters, the interface itself does not adjust properly when resizing the window; the table does not expand to fill the increased window size. We also take issue with the use of a drop-down list for such a massive number of organisms. This makes traversing the list tedious and difficult to find what the user is looking for. Button states also do not adjust to reflect the available actions the user can perform. For example, the "select all" button will remain active even when all pathways were already selected. Lastly, there are some elements which simply shouldn't exist at all. For example, there is a "File" button which does absolutely nothing when pressed.

4.1.2 Functionality

Our system contains every feature of KPGminer, as well as improvements and additional functionality. For example, the user can select organisms and pathways as they can in KPGminer, however, in our system the user is also able to search in real-time for organisms and pathways. Our KEGG Pathway Navigator can also let the user search by a prefix code or pathway ID, while KPGminer does not. We also include support for pathway descriptions from the KEGG database, while KPGminer does not despite having a column for it. Our system also allows users to configure options when downloading the pathways including exporting to .csv file as well as .gtm.

We also encountered bugs with some of the KPGminer functionality. For example, we noticed that certain pathway names were not extracted properly, as shown in figure 4. We also found issues with some pathway genes being extracted incorrectly as well when written to a file.

4.1.3 Performance

In terms of performance, our KEGG Pathway Navigator application outperforms KPGminer in full pathway retrieval time by nearly a half. By full pathway retrieval time, we mean the time it takes to extract all pathways for a given organism; in our test we used the human organism. The approximate time in seconds for both tools are shown in table 1, recorded over three trials each on a separate day. Caching was disabled on our system during these tests as that would be an unfair comparison, since caching reduces our retrieval time down to just a second.

Our performance gains are based on how we handle batched requests as discussed in section 3.1.5. KPGminer sends individual requests for each pathway, whereas we batch up to ten pathways into a single request.

Lastly, caching on our server provides a huge benefit in terms of performance. When requested data is cached, it leads to near instantaneous requests for organisms and pathways. Since the cache is shared it also grows as other users issue requests. This cuts down on the number of duplicate requests that the KEGG database needs to process, since we can cache results that would normally take a series of requests and cache the

Table 1. Comparing full path retrieval time (in seconds) of KPN and KPGminer

	KPN	KPGminer
Day 1	40s	85s
Day 2	44s	87s
Day 3	39s	91s
Avg.	41s	88s

entire result. This is one major advantage of choosing a web application as our platform of choice.

4.2 Request Limits and Parallelization

We have also explored the use of concurrent connections when requesting data from the KPGminer database. This is a future idea discussed in the previous work [2]. We have explored this idea for the performance gains of parallelizing our database requests, however, we found that very quickly we were being rate limited by the KEGG database. While their site has no mention of what the API rate limits are, we found that just having three and sometimes two concurrent connections opened resulted in 4xx server errors.

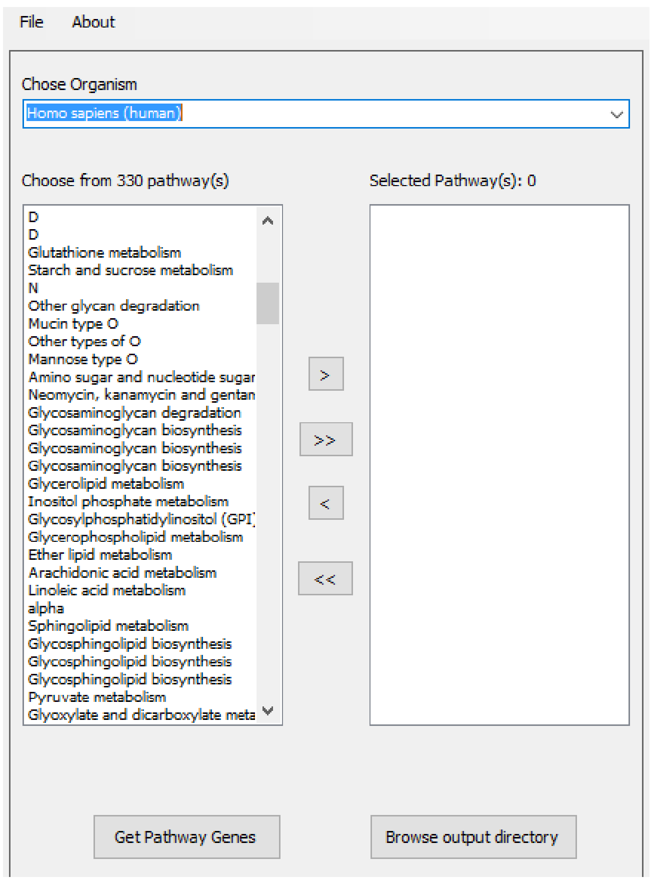
For these reasons, we have chosen not to use concurrent requests, as the rate limiting was too great and unpredictable. We have encountered no issue sending requests in serial at about 1 request per second. This is not as much of an issue, however, because we still are able to effectively parallelize up to ten requests by merging the requests into a single batch request as discussed in section 3.1.5.

5 Conclusion

We have presented KEGG Pathway Navigator, a modern, open-source, cross-platform web application for streamlining the retrieval of pathway genes from the KEGG Pathway database. We have discussed our approach to the technology, functionality and interface. We have seen the benefits of implementing our tool as a web application, such as shared caching and ease of access. We have shown how our system preserves all of the original KPGminer functionality and expands on their system by adding new features such as real-time search and configurable download options.

We have also shown how our interface conforms to modern UI practices and guidelines, such as responsive layouts and element states that accurately reflect the state of the application. We have looked at the exact implementation details regarding both our frontend and backend. We have shown the details behind our asynchronous requests and the batch optimizations we have made to improve the performance of our tool. We have compared our system to the existing KPGminer Windows executable, and found our system outperforms in design, functionality and performance.

For the future, we would like to expand on the number of configurable options when exporting to a file, to help researchers better tailor the results to their requirements. We also would like to explore extracting more information from the KEGG Pathway database that researchers might find useful. We would also like to expand beyond just the KEGG Pathway database and integrate other databases such as Wikipathways [6].



Chose Organism: Homo sapiens (human)

Choose from 330 pathway(s)

D
D
Glutathione metabolism
Starch and sucrose metabolism
N
Other glycan degradation
Mucin type O
Other types of O
Mannose type O
Amino sugar and nucleotide sugar
Neomycin, kanamycin and gentan
Glycosaminoglycan degradation
Glycosaminoglycan biosynthesis
Glycosaminoglycan biosynthesis
Glycosaminoglycan biosynthesis
Glycerolipid metabolism
Inositol phosphate metabolism
Glycosylphosphatidylinositol (GPI)
Glycerophospholipid metabolism
Ether lipid metabolism
Arachidonic acid metabolism
Linoleic acid metabolism
alpha
Sphingolipid metabolism
Glycosphingolipid biosynthesis
Glycosphingolipid biosynthesis
Glycosphingolipid biosynthesis
Pyruvate metabolism
Glyoxylate and dicarboxylate meta

Selected Pathway(s): 0

Fig. 4. Bug in KPGminer. Some of the pathway names are incorrect (e.g. "D" and "N"))

References

[1]Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25, 2000.

[2]AKM Azad. Kpgminer: A tool for retrieving pathway genes from kegg pathway database. *bioRxiv*, p. 416131, 2018.

[3]AKM Azad and Hyunju Lee. Voting-based cancer module identification by combining topological and data-driven properties. *PloS one*, 8(8):e70498, 2013.

[4]David Croft, Antonio Fabregat Mundo, Robin Haw, Marija Milacic, Joel Weiser, Guanming Wu, Michael Caudy, Phani Garapati, Marc Gillespie, Maulik R Kamdar, et al. The reactome pathway knowledgebase. *Nucleic acids research*, 42(D1):D472–D477, 2013.

[5]Minoru Kanehisa. The kegg database. In *âĽˇIn SilicoâĽ™Simulation of Biological Processes: Novartis Foundation Symposium 247*, volume 247, pp. 91–103. Wiley Online Library, 2002.

[6]Thomas Kelder, Martijn P van Iersel, Kristina Hanspers, Martina Kutmon, Bruce R Conklin, Chris T Evelo, and Alexander R Pico. Wikipathways: building research communities on biological pathways. *Nucleic acids research*, 40(D1):D1301–D1307, 2011.

[7]Aravind Subramanian, Pablo Tamayo, Vamsi K Mootha, Sayan Mukherjee, Benjamin L Ebert, Michael A Gillette, Amanda Paulovich, Scott L Pomeroy, Todd R Golub, Eric S Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 2005.