ARPHA Preprints

PENSOFT

Project Report

Author-formatted document posted on 30/05/2023

Published in a RIO article collection by decision of the collection editors.

DOI: https://doi.org/10.3897/arphapreprints.e107166

Deliverable D11.2 Search and link association services: A RESTful API, which will input a link/accession number and return a ranked list of neighbours links with a confidence score

Soulaine Theocharides, **Niki Kyriakopoulou**

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

Search and link association services: A RESTful API, which will input a link/accession number and return a ranked list of neighbours links with a confidence score

Deliverable D11.2

31 October 2022

Authors

Soulaine Theocharides[1, 2], Niki Kyriakopoulou[1, 2]

[1] *Naturalis Biodiversity Center, Leiden, Netherlands*
[2] *Distributed System of Scientific Collections - DiSSCo, Leiden, Netherlands*

BiCIKL

BIODIVERSITY COMMUNITY INTEGRATED KNOWLEDGE LIBRARY

This project receives funding from the European Union's Horizon 2020 Research and Innovation action under grant agreement No 101007492.

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

Start of the project:	May 2021
Duration:	36 months
Project coordinator:	Prof. Lyubomir Penev Pensoft Publishers
Deliverable title:	Search and link association services: A RESTful API, which will input a link/accession number and return a ranked list of neighbours links with a confidence score.
Deliverable n°:	D11.2
Nature of the deliverable:	Report
Dissemination level:	Public
WP responsible:	WP11
Lead beneficiary:	Naturalis Biodiversity Center
Citation:	Theocharides, S. & Kyriakopoulou, N. (2022). *Search and link association services: A RESTful API, which will input a link/accession number and return a ranked list of neighbours links with a confidence score.* Deliverable D11.2, EU Horizon 2020 BiCIKL Project, Grant Agreement No 101007492.
Due date of deliverable:	Month 18
Actual submission date:	31 October 2022

Deliverable status:

Version	Status	Date	Author(s)
3.0	Final	28 October 2022	Soulaine Theocharides, Niki Kyriakopoulou
2.0	Reviews submitted	15 October 2022	Quentin Groom, Lyubomir Penev
1.0	Final/Draft	6 October 2022	Soulaine Theocharides, Niki Kyriakopoulou
0.1	Initial Draft, APIs	5 October 2022	Soulaine Theocharides, Niki Kyriakopoulou

Table of contents

Summary 4

List of abbreviations 4

1. Introduction 5
    1.1. Background 5
    1.2. Scope 6

2. Endpoints 6
    2.1. Functional Endpoints 7
    2.2. Prediction Endpoints 7
    2.3. Observation-Only Endpoints 8

3. Classifier 9
    3.1. Training the classifier 10
        3.1.1. Integer Mapping 10
        3.1.2. Generating Non-Interaction Data 11
        3.1.3. Geographic Training Features 12
        3.1.4. Training on Other Features 13
    3.2. Weaknesses 13

4. Architecture 14
    4.1. Flask API 14
    4.2. Database 14
    4.3. Deployment 15

5. Recommendations and Future Work 15

6. Acknowledgements 16

7. References 17

8. Appendix: Wireframes 18

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

Summary

Work package 11 of the BiCIKL project involves developing software tools to support a FAIR experience for members of the biodiversity research community. The package overall focuses on Findability, by providing tools to search and answer questions, and Accessibility, through developing links across various biodiversity data sources and research tools. Task 11.2 specifically involves prediction of new links using machine learning.

We chose to demonstrate the functionality of machine learning link prediction with plant-pollinator interactions. This type of interaction was chosen due to the wealth of data available, particularly on the Global Biotic Interactions (GloBI) database, as well as this kind of interaction's ecological and economic significance. The result was a RESTful API capable of predicting plant-pollinator interactions among a predefined set of species. Predictions are made on-the-fly, at the time of the request. The GitHub repository for the API can be found here.

The API takes either a plant or a pollinator as inputs, and outputs potential matches based on a user-defined confidence score. The API's prediction is powered by a random forest classifier stored on disk. The classifier was trained on the taxonomic hierarchy of observed plant-pollinator pairs obtained from the GloBI database. When evaluating the likelihood of an interaction, the trained classifier looks at the taxonomic hierarchy of both the plant and pollinator and outputs a confidence score. What pairs are returned is determined by the minimum confidence score set by the user.

List of abbreviations

Amazon EC2	Amazon Elastic Compute Cloud
API	Application Programming Interface
BiCIKL	Biodiversity Community Integrated Knowledge Library
BLUE	Biotic Linkages United Explorer
DNS	Domain Name System
EU	European Union
FAIR	Findability, Accessibility, Interoperability, Reusability
GBIF	Global Biodiversity Information Facility
GloBI	Global Biotic Interactions
Gunicorn	Green Unicorn (a type of WSGI server)
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
SMOTE	Synthetic Minority Oversampling Technique
SQL	Structured Query Language
WSGI	Web Server Gateway Interface

1. Introduction

1.1. Background

The BiCIKL WP11 entails two main goals regarding biodiversity data that directly align with the FAIR guiding principles of Findable, Accessible, Interoperable and Reusable. More specifically, it aims to ensure Findability by enabling the search of relevant data as well as Accessibility in the form of linkages created between different data types and sourced from several biodiversity data infrastructures. In the same direction, Task 11.2 is consistent with these goals by providing end users with services that enable a) retrieval of information about possible links between data coming from scattered biodiversity resources, b) implementation of link prediction for the exploration and discovery of existing and new bi-directional linkages between data types, and c) curation of the predicted links through validation by the end users. Machine learning techniques are essential for the link prediction task which involves the identification of missing links and prediction of future links in networks.

Such networks are formed by complex biotic interactions between numerous players (species, individuals), as well as between organisms and their environment, namely, the ecological interaction networks which characterise an ecosystem and sustain life on earth. These networks often represent a particular function within an ecosystem such as pollination or parasitism. Biotic interactions can be positive or negative and take up several forms such as predation, commensalism, mutualism, and parasitism (Fraser *et al.* 2020). A well-studied type of mutualistic interaction is pollination (or plant-pollinator interaction), and this type of relationship was chosen as the proof of concept interaction for this project.

Pollination is of great importance since most crop plants grown around the world require pollination by animals. Moreover, flowering plants which are dependent on pollination provide environmental benefits such as clean air through carbon cycling/sequestration, purification of water and prevention of soil erosion through their roots. In addition, pollinators and plants are often recognised for their great cultural significance through their symbolism in various cultures, their use as food source, the use of plants in traditional medicine practices since prehistoric times, as well as natural or organic dyes extracted from natural resources such as insects, plants or parts of a plant.

Despite the ecological and economic importance of plant-pollinator relationships, there remains many unknowns about this subject. Listing all interactions, or estimating which ones may exist, is a tedious, time-consuming process (Strydom *et al.* (2021)). Machine learning technology has the potential to expedite this generation of knowledge, generating predicted links that experts can validate and invalidate according to their expertise.

Our starting point is making inferences about the types of biotic interactions based on taxonomic hierarchy. The presence of these pairwise biotic interactions has already been confirmed from the biodiversity literature. Inductive machine learning (specific to general) is employed to train our data based on general and simple rules defining pollination interactions (e.g., insects pollinate plants). Identification of species more likely to share similar interactions (or interacting groups of species) is needed.

The result was the Biotic Linkages United Explorer (BLUE) API. The BLUE API allows users to retrieve information about possible links (or interactions) between a pollinator and all the potential species interacting with it, based on a machine learning classifier trained on data from the GloBI database. A user - either machine or human - can input a taxon of interest into the API, and the API curates a list of predicted interactions. Once a possible interaction is

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

identified, then the probability of this link occurring between two species is calculated and presented in the form of a confidence score. Links with the highest confidence score are returned to the user.

1.2. Scope

This task delivered a RESTful API that supports an advanced FAIR experience for users. It is aligned with the goals of the WP11, focusing on Findability (search of taxon names) and Accessibility (creation of linkages). This service enables the discovery and display of bi-directional links between taxa, taxonomic hierarchies and geographic information.

Part of work package 11.2 discusses link validation, and outlines how the API will enable the curation of the predicted links via user validation; however, the current state of the API is “read-only.” The API is capable of predicting interactions based on a taxon of interest and presenting these predictions to the user, but the user can not validate these predictions. More specifications as to what it means to validate predictions is needed before this part of the task can be tackled. Further discussion on this subject can be found in Section 5: Recommendations and Future Work.

Currently, the API only supports the prediction of plant-pollinator relationships. Predictions are made on a taxon of interest provided by the user. If the taxon of interest is a plant, the API predicts potential pollinators; if the taxon of interest is a pollinator, the API predicts potential plants it pollinates. The API also provides some “observed” interactions, which are interactions obtained from the GloBI database. The plant-pollinator relationship was chosen as the proof of concept for this task due to its ecological significance and wealth of available data. Additional interactions, such as predation and parasitism, can be implemented by training a separate classifier.

In addition to the API, a set of wireframes (Appendix) was created with the Balsamiq software and constitute a visual guide that represents the skeletal framework of the BLUE API’s graphic interface. Additional functionalities presented in the following mockups are the graph representation of the results table that highlight the bi-directional links between taxa names, taxonomy and geographic information. Users will be able to validate newly predicted links by checking boxes next to each resulting pair. The new links will also be represented in a graph form.

This document gives a technical overview of the BLUE API. Section 2 describes the endpoints of the API. Section 3 delves into the classifier that powers the BLUE API’s linkage prediction, and describes the training process, data preprocessing, and how the classifier makes its predictions. Section 4 then outlines the other components of BLUE: The API, the database, and the deployment. Finally, recommendations and next steps are discussed in Section 5.

2. Endpoints

This section describes the available endpoints for the BLUE API. The API is available through the DNS blue.bicikl-project.eu/. The API uses GBIF Taxon IDs to identify unique species. In the future, it will be desirable to accept species names as well as taxon ids. This can be achieved by querying the GBIF taxonomic backbone in a layer between the user and the application.

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

2.1. Functional Endpoints

/hello

Confirms connection to API

/plants

Returns taxon_id of all the plants in the database that have one or more associated pollinator

/pollinators

Returns taxon_id of all pollinators in the database

2.2. Prediction Endpoints

/pollinatorOf/{taxon_id}

Body (Optional):

```
{
  "conf":0.5
}
```

Returns pollinators of the taxon of interest, **checking against all pollinators in the database**. Returned values include both observed interactions and predicted values; for the latter, a confidence value is given.

The optional parameter “conf” allows the user to set the minimum confidence threshold to return potential matches. The default value for “conf” is 0.5.

/pollinatedBy/{taxon_id}

Returns plants pollinated by the taxon of interest by classifying all entities in the database. Like the /pollinatorOf endpoint, this endpoint returns both observed and predicted values. This endpoint can take the same request body as /pollinatorOf as well.

/predict

Body (mandatory):

```
{
  "relation":"pollinates",
  "is_subject":true,
  "taxon_id": 1314881,
  "check": [2928234, 2964138, 2781074],
  "confidence":0.6
}
```

This endpoint accepts specific taxa to be checked against the taxon of interest, returning confidence values for the most likely taxa. A description of the body arguments is seen in the table (**Table 1**) below.

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

Table 1: *Body arguments for the /predict endpoint.*

Argument	Description	Example
relation	Interaction of interest. Currently, only “pollinates” is supported. More relationships are planned to be supported, such as “predatesOn” and “parasiteOf.”	“pollinates”
is_subject	Which side of the relationship the **taxon of interest is**[1] Interactions are conceptualised as: Subject X <interacts with> Target Y If this flag is set to true, BLUE will return targets of the relation (i.e. Ys). If the flag is false, it will return subjects of the interaction (i.e. Xs).	true
taxon_id	GBIF taxon id for the taxon of interest	1314881
check	List of taxon ids to validate described relationship with taxon of interest	[2928234, 2964138, 2781074]
confidence	Minimum confidence for returned predictions. Decimal value.	0.65

2.3. Observation-Only Endpoints

The following endpoints can be used to query observed interactions. They represent the possibility of growth for the BLUE API. With more classifiers trained, these endpoints can be modified in the future to also include predicted values

/predatorOf/{taxon_id}

Returns predators of taxon of interest. Taxon of interest is prey.

/predatedBy/{taxon_id}

Returns taxa predated by taxon of interest. Taxon of interest is the predator.

/parasitizedBy/{taxon_id}

Returns taxa parasitized by taxon of interest. Taxon of interest is the host.

/parasitizes/{taxon_id}

Returns taxa that the taxon of interest parasitizes. Taxon of interest is the parasite.

[1] Pollinates: If this flag is true, the API will look for plants the taxon of interest pollinates. If false, the API will look for pollinators of the taxon of interest.

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

3. Classifier

The classifier predicts how likely a pollinator-plant pair is to be a valid match based on taxonomic information. A high-level overview of the data flow is given in **Figure 1.** The classifier takes the taxonomic hierarchy of two taxa of interest and outputs a confidence score as a decimal.

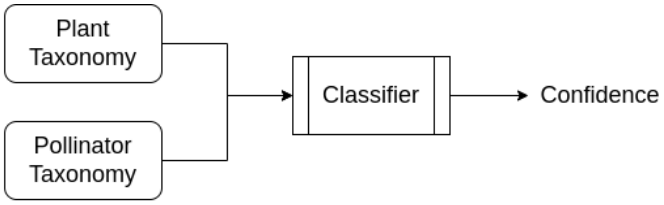


Figure 1: *The classifier evaluates a single relationship between a plant and a pollinator at a time and outputs a confidence score.*

When a query is made, the classifier is presented with a taxon of interest and a list of potential matches. Given this list, the classifier is able to identify the most likely valid pairs based on an established (user-defined) threshold. When making predictions, the classifier is given a table (**Table 2**), wherein each row represents a pollinator-plant pair's taxonomy.

Table 2: *Sample conceptual row-ise input and output of classifier. The classifier evaluates the likelihood that a taxon of interest, Rosa acicularis, is pollinated by three different pollinators. Note: for simplicity, some taxonomic levels have been omitted in this table.*

Plant Kingdom	Plant Species	Pollinator Kingdom	Pollinator Species	Output: Confidence
Plantae	Rosa acicularis	Animalia	Cortodera longicornis	0.99
Plantae	Rosa acicularis	Animalia	Apis mellifera	1.0
Plantae	Rosa acicularis	Animalia	Ophioplinthus brevirima	0.1

The classifier used is a SMOTE random forest classifier implemented using the Sci Kit Learn Python package. The random forest classifier was chosen because it trains quickly relative to other machine learning models and it performs well. See this blog post for an overview of how random forest classifiers work. Using SMOTE, or Synthetic Minority Oversampling Technique, allows the classifier to adapt better to highly skewed data (i.e. more data in one class than another). Data processing steps and the training of the classifier can be found on GitHub.

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

3.1. Training the classifier

3.1.1. Integer Mapping

The features used to train the classifier are: kingdom, phylum, order, family, genus, and species for both plants and pollinators (this is the same list of features required to classify an interaction).

In order to train the classifier and make predictions, string data (i.e. the taxonomic hierarchy) had to be transformed into integer data. This mapping is referenced when evaluating pollinator-plant pairs, transforming taxonomic data. Each taxonomic level is mapped independently. An example is demonstrated below (**Table 3**).

Table 3: *Example integer mapping of three plant species.*

Kingdom	Kingdom_int	Family	Family_int	Species	Species_int	Map
Plantae	0	Lamiaceae	0	Monarda fistulosa	0	[0, 0, 0]
Plantae	0	Rosaceae	1	Rosa acicularis	1	[0, 1, 1]
Plantae	0	Rosaceae	1	Rubus deliciosus	2	[0, 1, 2]

The kingdom is 0 for all examples because all entities come from the kingdom Plantae; the Lamiacae family is given integer 0, and Rosacae is given 1; and each species value is given its own integer. The entities themselves are identified by the combination of integer-mapped taxonomic levels. As such, [0,0,0] refers to *Monarda fitstulosa*, [0,1,1] refers to *Rosa acicularis*, and [0,1,2] refers to *Rosa deliciosus*. These integer arrays are what the classifier uses as input data. These labels hold no phylogenetic significance; they are simply used to map text data into a format the classifier can understand.

This integer mapping was generated while the training data are being preprocessed. The mapping was saved at that time. When the classifier evaluates a plant-pollinator pair, the taxonomic hierarchy of both species is transformed to integers using this mapping. The process of integer mapping occurs right before data are classified. **Figure 2** gives a modified version of **Figure 1** which indicates when this process takes place.

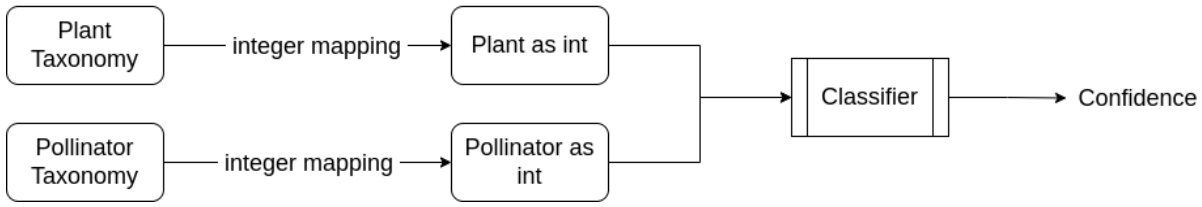


Figure 2: *Plant and pollinator taxonomic hierarchy are transformed into integers before being presented to the classifier.*

3.1.2. Generating Non-Interaction Data

Training data was obtained from the GloBI database. Interactions from this database were coded as "Class 1", or true values. However, in order to train a machine learning classifier, "Class 0" or false values, representing non-interaction data, also need to be present in the training data. Non-interaction data were constructed by randomly shuffling observed plant-pollinator pairs.

This process is illustrated in the following (colloquial names have been used for clarity). In this example, roses are pollinated by honeybees, daisies are pollinated by butterflies, and cacti are pollinated by hummingbirds. These are true interactions, noted as Class 1. Relations outside these observed interactions may be marked as false, or Class 0 (e.g. roses pollinated by hummingbirds).

The ratio of Class 1 and Class 0 present in the training data should reflect the desired output. If the classifier is trained with 50% accurate data and 50% falsified data, it will expect more positive results than if it were trained on 1% true data and 99% falsified data. The ideal ratio should be subject to testing and the design of the overall BiCIKL infrastructure. More Class 1s will result in more predicted relationships, potentially leading to false positives. On the other hand, more Class 0s may lead to the classifier missing interactions while preventing false positives.

We chose to train the classifier on a very skewed ratio of Class 0 to Class 1 to prevent false positives. Class 0s were obtained by matching each pollinator with each plant it did not pollinate, then removing 30% of these rows at random. Determining exactly which relations can definitively be marked as Class 0 is worth exploring. As Strydom *et al.* (2021) put it, "it is difficult to distinguish between a true negative (where the two species never interact) and a false negative (where two species interact but have not been observed doing so)".

The following table (**Table 4**) illustrates how Class 0s were generated.

Table 4: *Generating non-interaction data to train the classifier (some of them are omitted, indicated by a strikethrough).*

Plant Species	Pollinator Species	Class
Rose	Honeybee	1
Daisy	Butterfly	1
Cactus	Hummingbird	1
~~Rose~~	~~Butterfly~~	~~0~~
Rose	Hummingbird	0
Daisy	Honeybee	0
Daisy	Spider	0

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

~~Cactus~~	~~Honeybee~~	~~0~~
Cactus	Butterfly	0

3.1.3. Geographic Training Features

Originally, the classifier was trained on GBIF occurrence data on country level as well as taxonomic levels, but this proved challenging to implement. When training the classifier on country data, there is a separate row in the training data for each country a species occurs in. These rows are considered independently even though they contain members of the same species.

In the following example (**Table 5**), *Rosa acicularis* is found in France, the Netherlands, Belgium, and Canada; and *Bombus flavifrons* is found in Canada and the United States. In order to assess the relationship between these two species, the classifier needs to compare each *combination* of countries. Now, instead of validating a single row, the classifier is validating eight. This problem grows rapidly when classifying species that occur in several countries.

Table 5: *Output data showing the evaluation of the relationship between Rosa acicularis and Bombus flavifrons using country data. The highlighted rows represent interactions with high probability of occurrence (above a pre-specified threshold).*

Plant Species	Plant Country	Pollinator Species	Pollinator Country	Output: Confidence
R. acicularis	France	B. flavifrons	United States	0.2
R. acicularis	Netherlands	B. flavifrons	United States	0.1
R. acicularis	Belgium	B. flavifrons	United States	0.3
R. acicularis	Canada	B. flavifrons	United States	0.85
R. acicularis	France	B. flavifrons	Canada	0.1
R. acicularis	Netherlands	B. flavifrons	Canada	0.1
R. acicularis	Belgium	B. flavifrons	Canada	0.2
R. acicularis	Canada	B. flavifrons	Canada	1.0

There are other issues with training using country data. For instance larger countries may introduce some ambiguity. There is ultimately less information about a species observed in Canada (a country with an area of 9,985 million km^2) and a species observed in Aruba (180 km^2). This leads to inconsistency of the importance of this particular feature between different species, which can lead to undesirable behaviour. Country data may also be incomplete if a training set does not contain all countries of occurrence.

For the reasons outlined above, country data was ultimately omitted from training data. In testing, we did not find a significant drop in performance when country data was omitted.

Using geographic data may be more useful if presented in a concise and precise numeric format (e.g. features that describe a species' range) instead of country of occurrence.

3.1.4. Training on Other Features

Taxonomic hierarchy was chosen to train the classifier on for two reasons. First, taxonomic structure is consistent across each species, and it can be easily extracted from authoritative databases. Second, taxonomic hierarchy provides some indirect insight into morphology. Similar morphological traits are often used to group taxa in higher taxonomic levels with a common ancestry. While taxonomic hierarchy does not quantify the value of these traits, it does group similar species together, allowing the classifier to identify patterns.

However, taxonomy is not a fixed field of study (Thiele *et al. (*2021)). Taxonomic hierarchies change frequently as new information, such as genetic sequences, is uncovered. A classifier trained solely on taxonomy will be very sensitive to taxonomic changes, and the classifier's performance may deteriorate over time. Training on additional features, such as target morphology, habitat, type of behaviour, trophic level etc., should be added to yield more accurate results overall and provide resilience against taxonomic changes.

Thiele *et al. (2021)* acknowledge that including morphological traits does pose challenges. First, morphological traits that are most important to extract (and if key traits differ between taxonomic groups) must be identified. In addition, extracting those traits from literature and other resources poses an even greater challenge. However, these challenges also present opportunities for creating more linkages within BiCIKL partners and other work packages, such as WP 11.3: Passage retrieval.

Another approach would be the use of co-occurrence data as this has been implemented by Strydom *et al.* (2021). The lack of available species-level data can be counterbalanced by combining observed interactions, co-occurrence data and deep neural networks to infer novel biotic interactions. A necessary condition in order for an interaction to exist between two species is for them to occur together, both spatially and remporally. For example, a parasite must at some point be co-located with its host, a plant with its pollinator, a predator with its prey etc. As a result, distribution data coming from species sampling, observations, and preserved specimens can be used to infer potential inter-species interactions.

3.2. Weaknesses

The greatest weakness of the classifier is the encoding of categorical levels to integers, the "integer mapping" described in Section 3.1.1 The classifier is trained on taxonomic levels mapped to integers, not the taxonomic levels themselves. This mapping is saved and referenced when the classifier is tasked with making predictions. Since the map is created using training data, only taxonomic levels present in the training data can be mapped to an integer array for classification. While the classifier performs well with taxa it has seen already, it is completely unable to handle new ones.

A less rigid feature transformation strategy may be helpful in combating this issue. For instance, hashing could be used to transform a string (taxonomic levels) into a fixed-length integer. Instead of relying on a mapping defined by the training data, a hashing algorithm can be applied to any string. The MD5 hashing algorithm, which transforms data into a 128-bit

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

value, could be used for such a use case. This hashing could be used to transform data from a taxon the classifier has not seen before into something ingestible.

However, even if the new taxon is capable of being processed by the classifier, it may result in unpredictable classification. This is a documented issue with Random Forest Classifiers. As *Au (2018)* observed, when categorical levels are present in a population but absent in a training set, the model can behave unpredictably. To avoid this scenario, two solutions present themselves. First, presenting the classifier with a wider range of species data may be a short term solution. Moving forward, however, it may be desirable to implement a new kind of classifier. Work has been done on deep neural networks in predicting species interaction (*Hirn et. al 2022*, *Pichler et. al 2019*), and this may be a tool more suited for this project.

4. Architecture

This section gives an overview of the primary components of BLUE. In addition to the classifier, which is stored locally as a joblib serialisation object, BLUE consists of a Flask API and a PostgreSQL database. Here, these components are presented, along with a description of the API's deployment stack.

4.1. Flask API

The BLUE API is a RESTful API built using the Flask framework. It has several endpoints, described in Section 2, and is powered by the trained classifier described in Section 3. The API's function is to receive input from the user, retrieve taxonomic hierarchy for all potential matches, present these matches to the classifier, and process the classifier's output. The trained classifier is stored locally as a joblib serialisation object. When a request from the user is received, the API loads the classifier, and the classifier then evaluates the given plant-pollinator pairs.

The API interfaces with the database using the SQLAlchemy toolkit. Upon receiving a request, the API retrieves taxonomic information about the taxon of interest from the database. If the user does not specify a list of taxa to check (i.e. using the /predict endpoint), the API will check all appropriate taxa stored in the database. It evaluates each pair, and returns matches with a confidence level above a specified threshold.

"Appropriate taxa" are determined by looking at the interaction type and the phylum the taxon of interest belongs to. A list of interaction rules was developed from literature to narrow down the data supplied to the classifier. Using the interaction rules, the program is able to select only species belonging to likely phyla and classify those species, instead of all possible species in the database. Subsetting at the phylum level is still broad enough to allow the classifier to make meaningful decisions while omitting ecologically impossible matches (e.g. a plant pollinating a mollusc or a vertebrate parasitizing a bacterium).

4.2. Database

Data are stored in a PostgreSQL relational database hosted on Amazon Web Services (**Figure 3**). Species interactions and observations were obtained from the Global Biotic Interactions

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

and GBIF databases respectively. The integer mapping required to transform taxon data into a form the classifier can process[2] is also stored in the database, in the classifier_tools table.

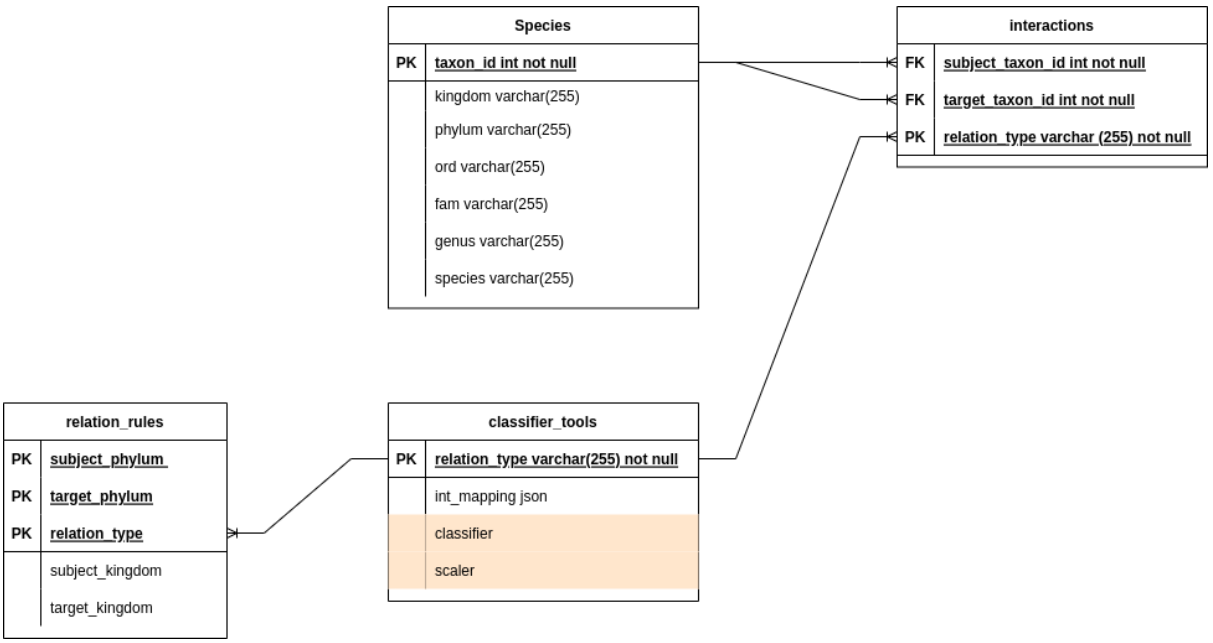


Figure 3: *Database schema for BLUE API.*

What is notably missing from this database is the classifier itself and the scaler used as a preprocessing step (denoted in yellow). Relational databases are not well-suited for storing unstructured data such as a machine learning model. The classifier is currently stored locally. A more desirable solution would be to store the classifier in an object storage facility, such as Amazon's S3 service. In the Postgres database, there should be a pointer to the S3 object storage. This table could be very useful for managing multiple classifiers for different interactions.

4.3. Deployment

The BLUE API is run on a Gunicorn WSGI server, which is behind an nginx reverse-proxy. The nginx reverse-proxy is a dedicated HTTP server that handles incoming requests securely for Gunicorn. The nginx reverse-proxy accepts requests at port 80 and forwards them internally to the Gunicorn server, which is listening at port 5000. The entire setup is dockerized into two images (one for Flask and Gunicorn, and one for nginx) and deployed on an Amazon Web Service EC2 server.

5. Recommendations and Future Work

The BLUE API is a prototype that uses a random forest classifier to evaluate potential pollinator-plant interactions. This classifier is specific to this type of interaction. Other

[2] See Section 3.1.1

ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

interactions, such as parasitic relationships or predatory relationships, can be supported with the addition of more classifiers. Depending on the number of interactions of interest, this may prove to be unfeasible.

It is likely that a random forest classifier, or many classical machine learning techniques, is not sophisticated enough to handle linkage prediction reliably or at the scale that BiCIKL envisions. On the other hand, deep learning techniques, such as transformers used by T11.4, through the use of neural networks, have shown promise in mapping ecological interactions (*Borowiec et. al (2021)*, *Strydom et al (2021)*, *Christin et. al (2019)*). A comprehensive literature review into the current state of the field of machine learning, deep learning, and linkage prediction before continuing with this work package is highly recommended.

More work needs to be done in terms of the user interface and allowing users to validate existing and newly predicted links. Specifically, a trust model must be developed through conversations with the community to determine the impact of validated links. Who can validate links, are some users' inputs more valued than others, how should validated links impact the underlying machine learning backbone (if at all)? Before these questions are answered, meaningful work on this aspect of the work package will be stalled.

6. Acknowledgements

We would like to acknowledge Laurens Hogeweg for his work on the random forest classifier.

7. References

Au, T. (2018). Random forests, decision trees, and categorical predictors: The “absent levels” problem. *Journal of Machine Learning Research*, 19, 1-30. https://doi.org/10.48550/arXiv.1706.03492

Borowiec, M. L., Dikow, R. B., Frandsen, P. B., McKeeken, A., Valentini, G. & White, A. E. (2021). Deep learning as a tool for ecology and evolution. *EcoEvoRxiv*. https://doi.org/10.1111/2041-210X.13901

Christin, S., Hervet, E. & Lecomte, N. (2019). Applications for deep learning in ecology. *Methods in Ecology and Evolution*, 10(*10*), 1632-1644. https://doi.org/10.1111/2041-210X.13256

Fraser, D., Soul, L. C., Tóth, A. B., Balk, M. A., Eronen, J. T., Pineda-Munoz, S., Shupinski, A. B., Villaseñor, A., Barr, W. A., Behrensmeyer, A. K., Du, A., Faith, J. T., Gotelli, N. J., Graves, G. R., Jukar, A. M., Looy, C. V., Miller, J. H., Potts, R. & Lyons, S. K. (2020). Investigating biotic interactions in deep time. Trends in Ecology & Evolution, 36(*1*), 61-75. https://doi.org/10.1016/j.tree.2020.09.001

Hirn, J., García, J. E., Montesinos-Navarro, A., Sánchez-Martín, R., Sanz, V. & Verdú, M. (2022). A deep Generative Artificial Intelligence system to predict species coexistence patterns. *Methods in Ecology and Evolution*, 13(5), 1052-1061. https://doi.org/10.1111/2041-210X.13827

Pichler, M., Boreux, V., Klein, A.-M., Schleuning, M. & Hartig, F. (2019). Machine learning algorithms to infer trait-matching and predict species interactions in ecological networks. *Methods in Ecology and Evolution*, 11(*2*), 281-293. https://doi.org/10.1111/2041-210X.13329

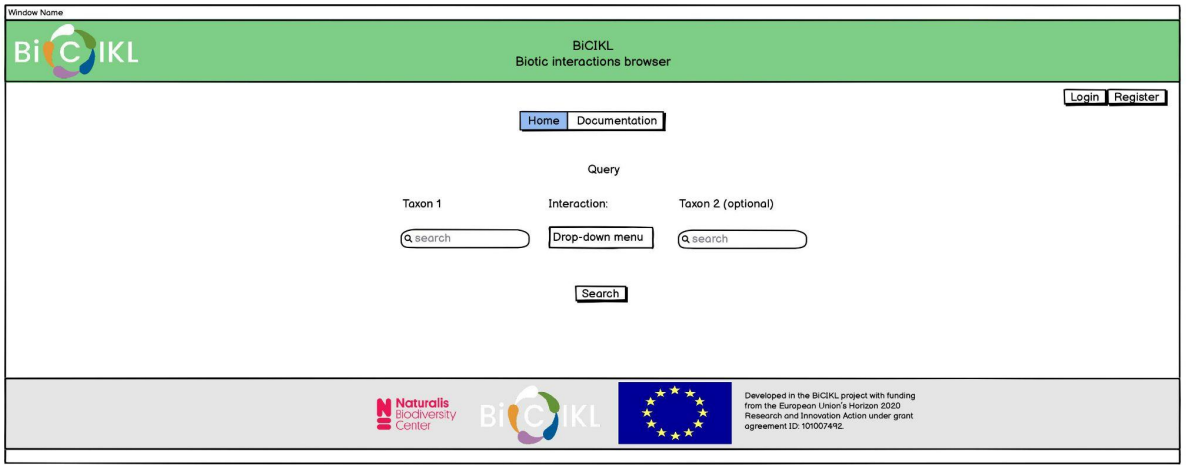
Strydom, T., Catchen, M. D., Banville, F., Caron, D., Dansereau, G., Desjardins-Proulx, P., Forero-Muñoz, N. R., Higino, G., Mercier, B., Gonzalez, A., Gravel, D., Pollock, L. & Poisot, T. (2021). A roadmap towards predicting species interaction networks (across space and time). *Philosophical Transactions of the Royal Society B*, 376(*1837*). https://doi.org/10.1098/rstb.2021.0063

Thiele, K. R., Conix, S., Pyle, R. L., Barik, S. K., Christidis, L., Costello, M. J., van Dijk, P. P., Kirk, P., Lien, A., Thomson, S. A., Zachos, F. E., Zhang, Z.-Q., Garnett, S. T. (2021). Towards a global list of accepted species I. Why taxonomists sometimes disagree, and why this matters. Organisms Diversity and Evolution, 21, 615-622. https://doi.org/10.1007/s13127-021-00495-y

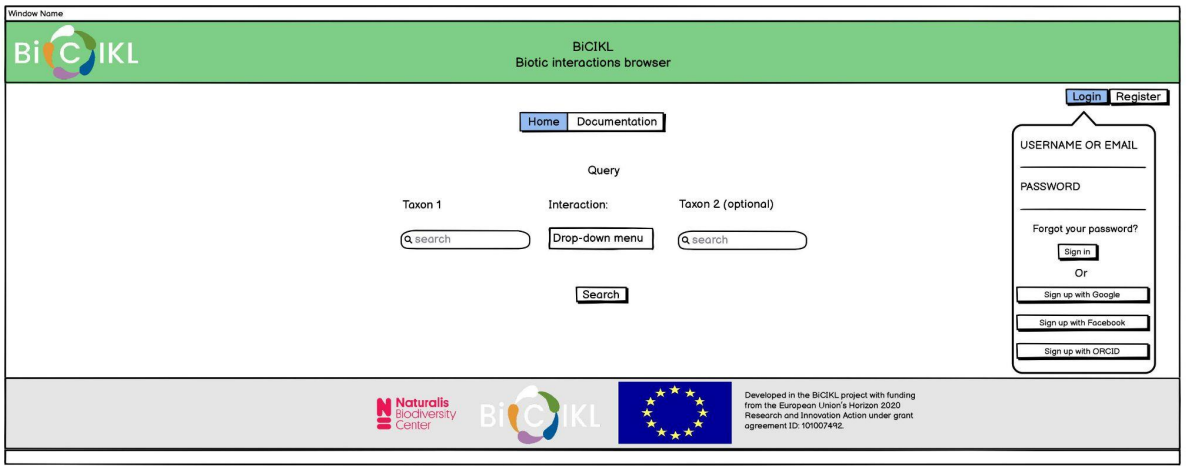
ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

8. Appendix: Wireframes

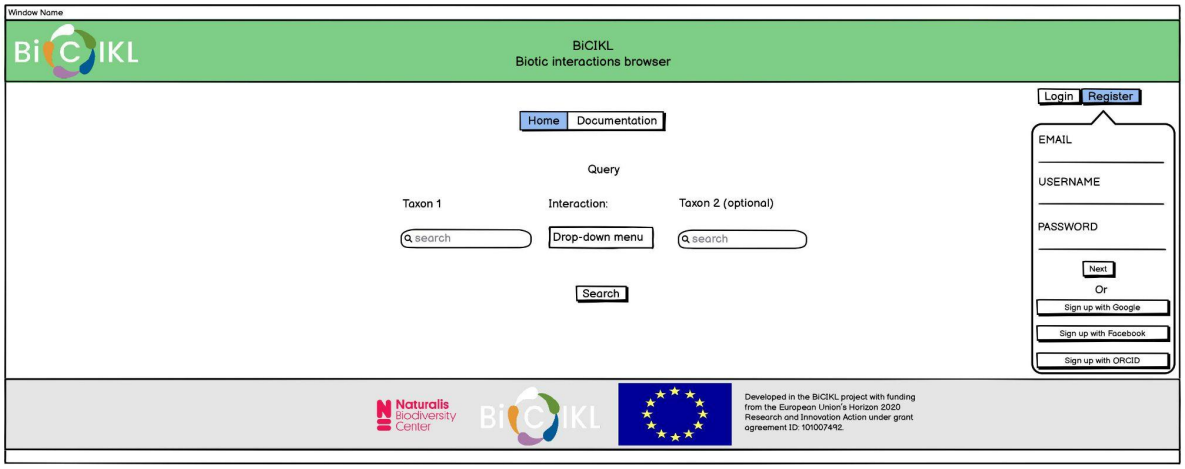
The first wireframe depicts the **Home** page of our API. The search engine along with a drop-down menu of possible interactions are depicted.



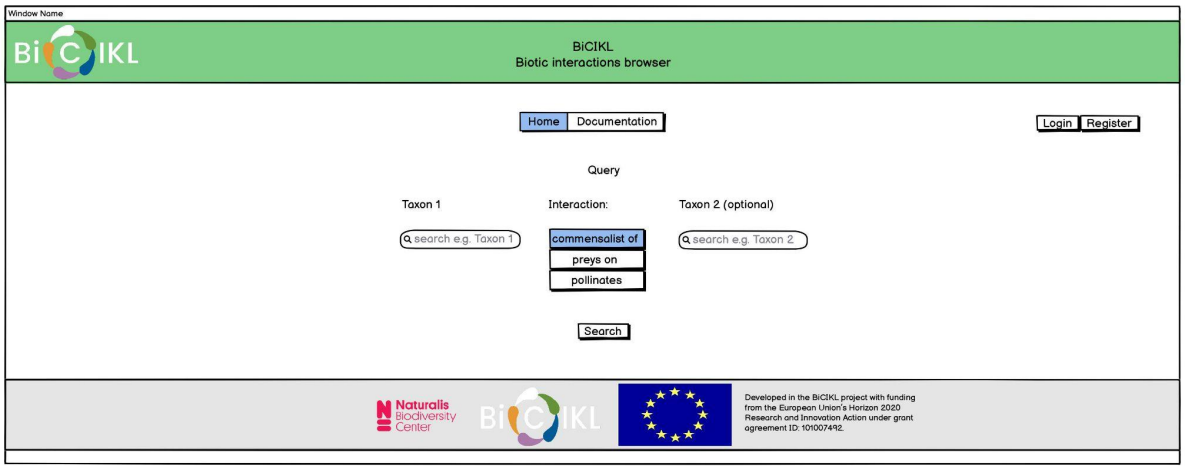
Users that want to validate predicted links have to log in. Registration is required and can be done by registering via Google, Facebook or ORCID.

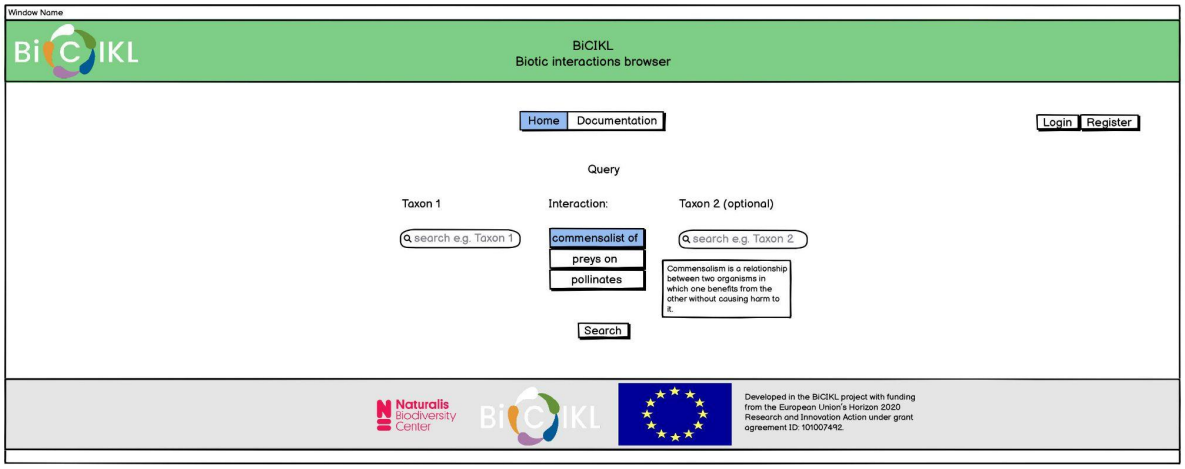


ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

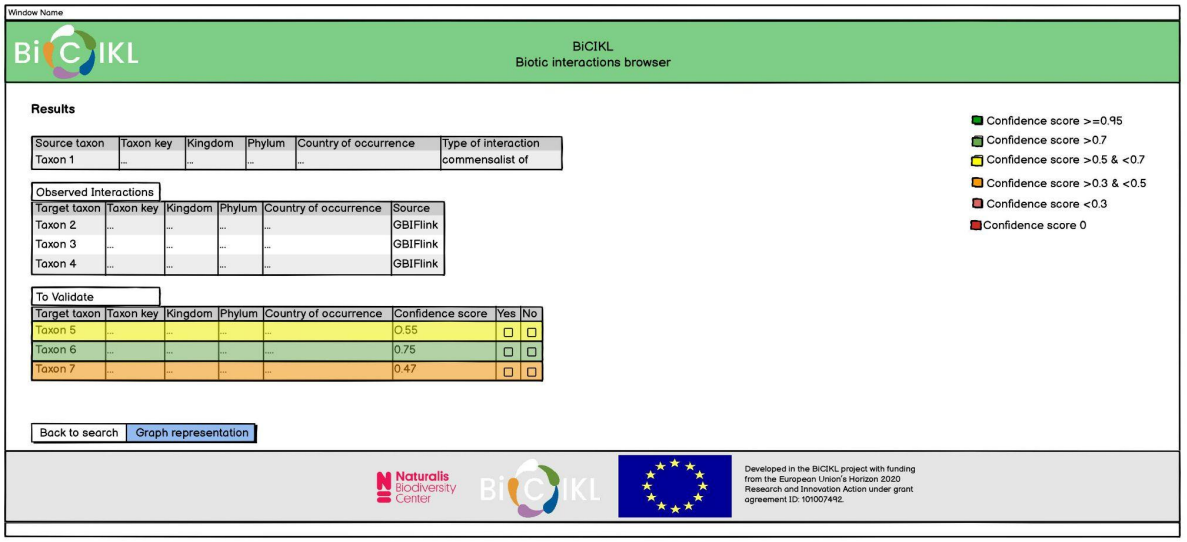


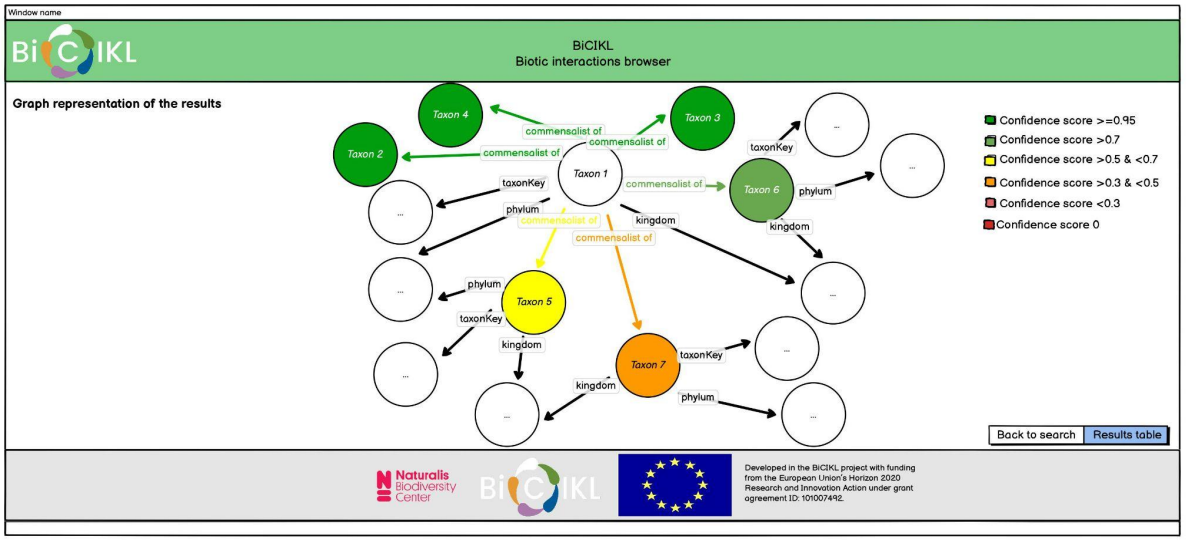
Users can input a taxon name on the left part of the search engine. A list of possible interactions appears. Hovering over each one of the interactions, a new window appears with the definition of the interaction. Selecting the desired interaction results into a table.





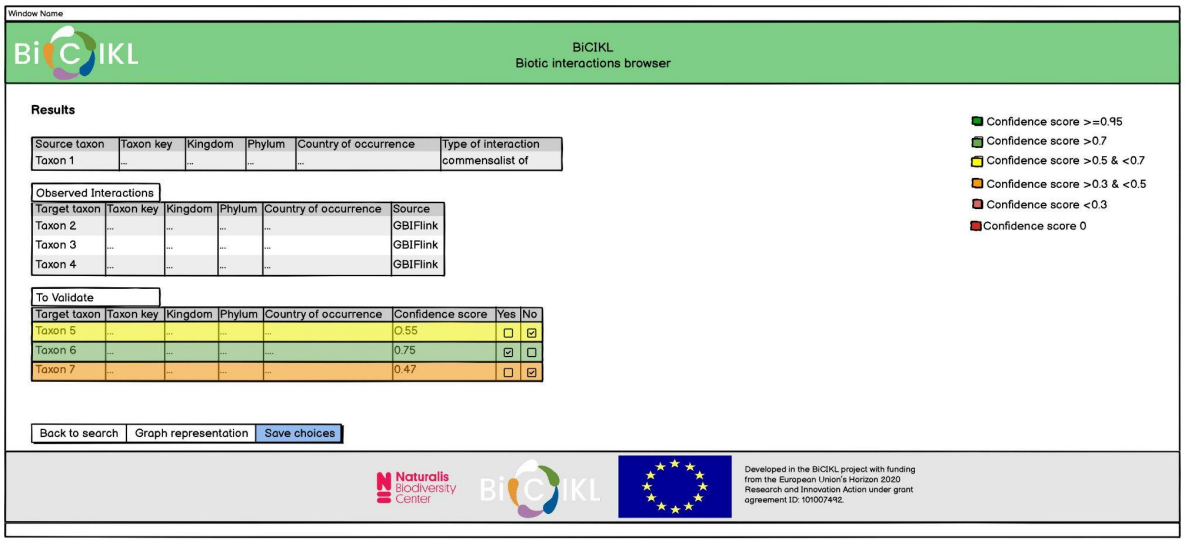
The first wireframe highlights the results tables. The input taxon is in the first table. The second table depicts the observed interactions based on GloBI. The last table contains the interactions predicted by the random forest classifier. Taxonomy and geographic information are included and enable the user to validate the newly predicted relationships. Different confidence scores are assigned different colours. The last table contains checkboxes for the registered users to validate the interactions. The graph representation tab can be used to present the results tables in the form of a knowledge graph.

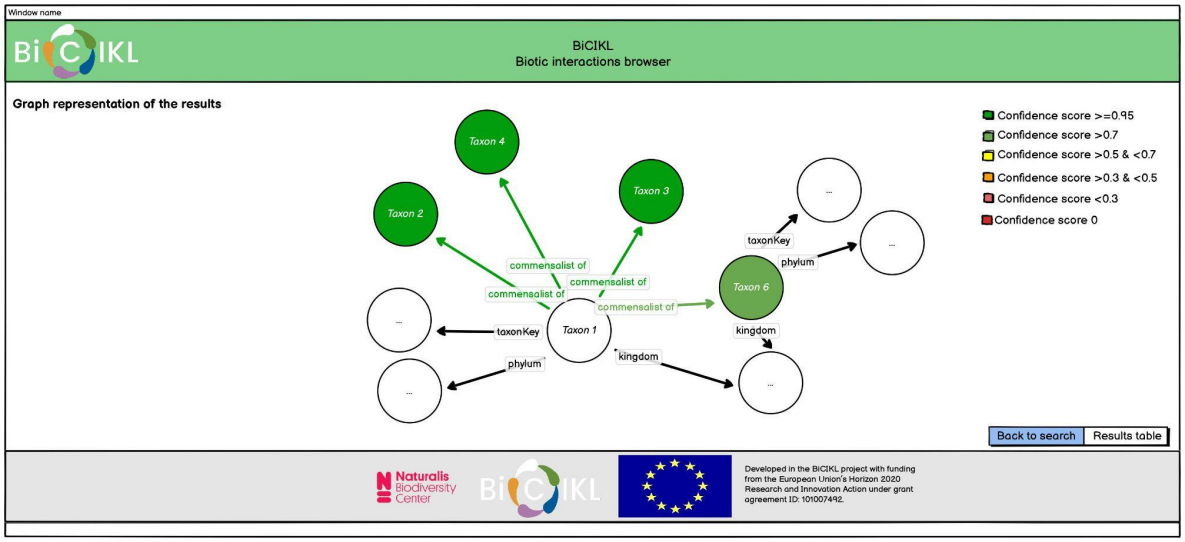




ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

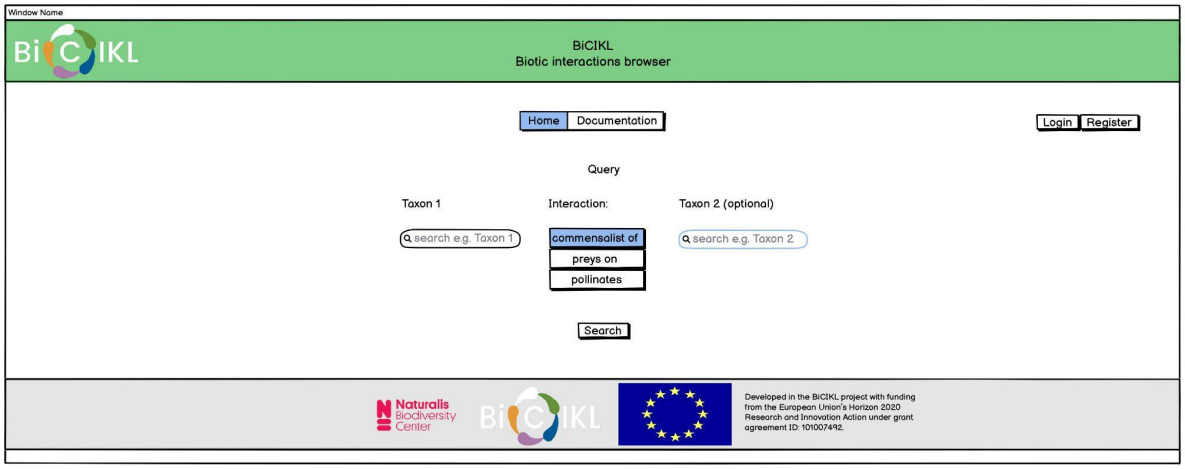
Users can check the boxes of the interactions they want to validate and save their choices in order for the results to be stored. By saving their choices, the newly validated interactions can be represented as a graph, with the rest of the non-validated relationships omitted.

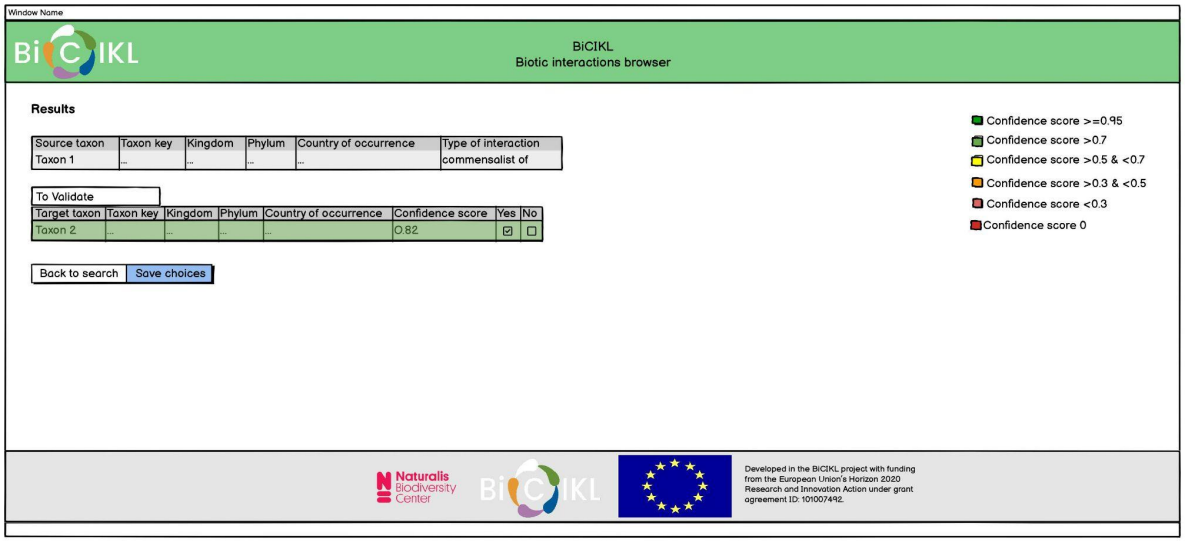


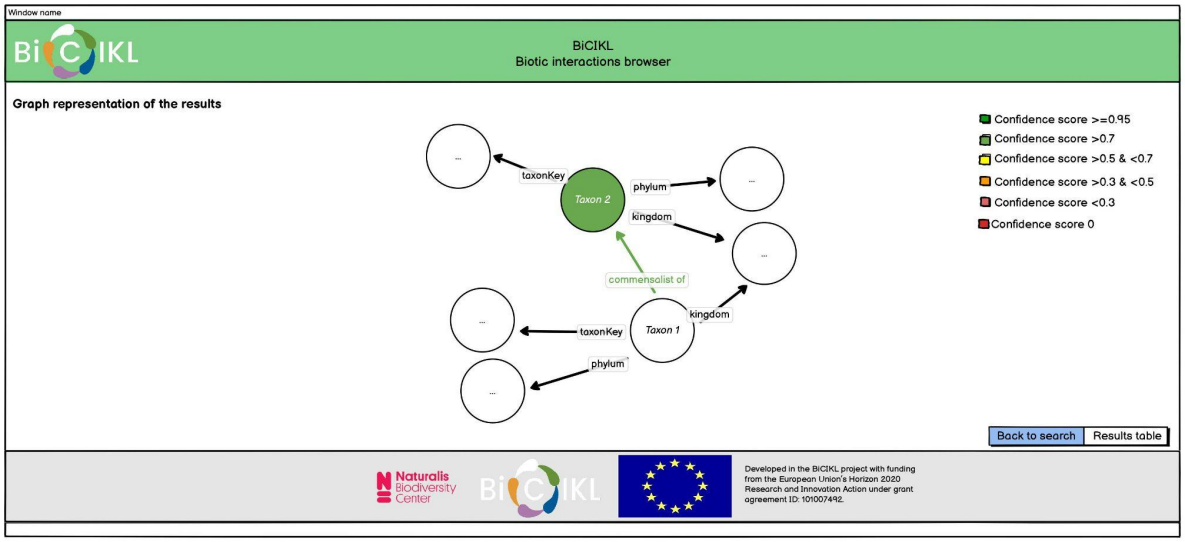


ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

Another possible type of search is inputting taxon names in both search engines. The results tables will only contain Taxon 1 and Taxon 2 along with the validation checkbox and the representation of results in a knowledge graph form.







ARPHA Preprints *Author-formatted document posted on 30/05/2023.* DOI: https://doi.org/10.3897/arphapreprints.e107166

Checking the **Documentation** tab brings us to the last part of the graphic interface where information about the functionalities of the API can be found.