Kestrel Taxonomy Finder

Version 1.0
03/04/2019

Shawn.M.Rupp@asu.edu

Contents

New in v1.0
Rewritten in Go
Searches concurrently
Removed common/scientific name considerations
Added white box tests

Introduction

Kestrel is a program for resolving species' common names and synonyms with "official" scientific names and extracting taxonomies from internet databases (i.e. EOL, NCBI, IUCN, ITIS, Wikipedia). It is meant to automate this process as much as possible, although, depending on your input data, some manual curation may still be required.

Dependencies:
Go 1.11+
Xvfb
Google Chrome
Active internet connection

Note: Kestrel is written and tested on a Linux platform, so there is no guarantee that it will work on Mac or Windows without troubleshooting. Additionally, the following instructions are written for Linux and may require some modification for different operating systems. (I would love to provide further support for this program, but unfortunately time and funding constraints dictate otherwise.)

Installation

Go
Kestrel has been re-written in Go to dramatically improve performance and simplify installation. Follow the instructions here to install and configure Go.

Xvfb
Selenium (which will be installed by the installation script) requires Xvfb to run. To install on Linux:

```
sudo apt-get install xvfb
```

Kestrel
Download the git repository, change into the directory, and run the install script.

```
git clone https://github.com/icwells/Kestrel.git
cd Kestrel/
./install.sh all
```

Testing the Installation
Once you have installed the program and its dependencies, you may with to run the test script:

```
./test.sh
```

If everything is properly configured, it should not throw any errors.

Getting Started

EOL API Key
Kestrel queries the Encyclopedia of Life heavily, so you will need to generate an api key. To do so, create and EOL profile if you do not already have one. Sign in, go to your profile, and click “edit my profile.” Next, generate an API key in the box on the right side. Once you generate the API key, copy it and paste it into example-API.txt on the same line as “EOL=”. Finally, change the file name to API.txt (otherwise it will be erased if you run “git pull”).

NCBI API Key
As of 2018, NCBI requires an API key for any extensive searching. Create an NCBI account, or sign in, and click on your user name in the upper right hand corner of the screen. This will take you to your settings page. The fourth box down will allow you to generate an API key. Copy this key and paste it after “NCBI=” in API.txt.

IUCN API Token
Kestrel also optionally queries the IUCN Red List database. If you wish to request a token, fill out this form and submit it. Once you receive your API token, paste it after “IUCN=” in API.txt. It can take several days for approval, so Kestrel can run without this API key. Fortunately, IUCN results can still be found via Kestrel’s Google search.

Running Kestrel

Assuming you have a functional internet connection, you are ready to run the program. Prior to running a Kestrel search, target terms must be extracted from an input file. Terms which are unlikely to produce a result will be filtered out and , and the remaining terms will be formatted so they are more likely to produce a match. These terms are then used as input in the second stage to search for taxonomies. This is done as a separate step to allow the user to correct any of the formatted or rejected terms if necessary.

To run Kestrel, you must open a terminal and change into ~/Kestrel/bin. This directory contains the API keys and the kestrel binary.

Extracting Search Terms
The first step is to extract, filter, and format the query terms. To extract terms, change into Kestrel/bin/ and enter:

```
./kestrel extract -c <column_number> -i <input_file>
     -o <output_file>
```

The input file should be either a csv, tsv, or txt (either single column or tab delimited) file containing the scientific or common names of species of interest. The -c flag signifies the column number where the names are located. This is a 0-based integer, so if the names are in the first column specify “-c 0”. The program will extract unique names from this column. It will only search for a specific name once to save time and avoid burdening servers. There is no default value since it is very easy to forget this parameter and it is better to run into an error up-front.

Queries will be rejected if there is an indication of uncertainty (i.e. a question mark, the words “not” or “unkown,” …) or any indication that the query is a hybrid animal (which will not have a taxonomy). If any other punctuation is detected, Kestrel will attempt to format the name so it has the best chance of returning a match (e.g. “&” are replaced with “and,” items in parentheses or quotation marks are removed, …). After explicit examination, any remaining punctuation (except periods, apostrophes, and dashes) and any numbers are removed. If the resulting term is less than 3 characters long, it is rejected. Lastly, spaces and apostrophes will be percent encoded immediately prior to searching (in the next step) to keep the results of this step readable.

Formatted search terms will be written to the given output file along with the unformatted name. Any rejected names will be written to “KestrelRejected.csv” with the reason it was rejected. Since there is no easy way to predict exactly what might be in the input file, terms in the output file may be examined and edited if needed. Rejected terms may also be manually formatted and pasted into the output file. Depending on the quality of the input, this could greatly improve the quality of the search output and will be less time consuming than editing the original input file.

Previous versions of Kestrel used Python’s Natural Language Processing Toolkit to sort search terms into common and scientific names. Go does not yet have an equivalent package to perform this step, so, since it seemed to have little effect on output, it has been left out of this version.

Searching for Taxonomies

Kestrel starts the search step by reading names from any existing output files before reading names from the input file. If the given output file already exists, it will append output to this file which allows you to resume if the program is interrupted.

Previous versions allowed for this stage to be run on multiple threads. Go is capable of spawning thousands of concurrent searches with no noticeable performance drop, so it will perform all of the searches concurrently. In order to comply with API limits, however, it must submit the terms in batches, but the evaluation steps may all be carried out at the same time.

API-based Search
To search for matches to the formatted search terms (whether or not they were manually checked), Kestrel will read the extracted names and formatted search terms from the input file and will search for matches with one thread per search term. If multiple input queries have the same formatted search term (i.e. “Dog,” “dog,” and “DOG” will all be formatted to “Dog”), the term is only searched once but output will be written for each input query.

```
./kestrel search -i <input_file> -o <output_file>
```

For common names, it will search EOL, NCBI, IUCN, and Wikipedia. If two or more databases return a taxonomy, the consensus match will be written to the output file. If there is only one match, it will be selected. If there is no match and the term has more than one word, Kestrel will remove the first word (the last word of a common name is usually the noun) and repeat the above process until a match is found or there is only one word left.

Google Search
If no match is found using the API-based search, Kestrel will immediately submit the term for a Google search using Selenium. Since it can be impossible to predict the format of a page returned by a Google search, this step is reserved for terms which could not be resolved using simpler means.

Kestrel will search for the given term and the word "taxonomy." It creates a list of the urls found in the search results and iterates through them. If a url is hosted on Wikipedia or ITIS (which are the most common results with complete taxonomies), the program will search that page for taxonomy data. Once the first complete match is found, it is appended to the output file. If a match is not found, the term is recorded in KestrelMissed.csv.

Output
For names with an identified taxonomy, output will be written to the given output file in csv format. The query name will be in the first column with the the formatted search term in the second, followed by a column for each taxonomy field. There is one column for each database that Kestrel queries and urls from accepted hits will be recorded in the source database's column. This identifies the source of the data and allows it to be accessed again. Any names for which a taxonomy is not identified will be written to the KestrelMissed.csv file, which will be located in the same directory as the output file.

Checking Output
It is strongly recommended that you manually inspect Kestrel's search output. Kestrel relies on the abilities of third party search algorithms and further error checking would require a much more complicated program.

The check command can be used to help with manual curation. The "-r" flag can given to supply an existing taxonomy file that has already been curated (it must be in the same format as Kestrel's search output, but it doesn't really matter where it came from). It will overwrite a non-curated taxonomy with the reference taxonomy if matching search terms are found, and can be used on the output of kestrel –extract to reduce the overall number of terms that must be searched.

If the -r flag is not given, Kestrel will compare the formatted search term with the species name. Queries which have matching names will be written to ".passed.csv" while the remaining queries will be written to ".failed.csv" for manual curation. Depending on the number of scientific names, this can greatly reduce time spent checking Kestrel output.

Commands

```
./kestrel {command} {options}
```

help	Show help.
version	Prints version info and exits.
extract	Extracts and filters input names.
search	Searches for taxonomy matches to extracted names.
check	Identifies search results with matching search terms and scientific names to streamline manual curration.
merge	Merges search results with source file.

Extract
Extracts and filters input names.

```
./kestrel extract -c <column_number> -i <input_file> -o <output_file>
```

-i, --infile=INFILE	Path to input file.
-o, --outfile=OUTFILE	Path to output csv file.
-c, --column=COLUMN	Column containing species names (integer starting from 0).

Search
Searches for taxonomy matches to extracted names.

```
./kestrel search -i <input_file> -o <output_file>
```

-i, --infile=INFILE	Path to input file.
-o, --outfile=OUTFILE	Path to output csv file.

Check
Identifies search results with matching search terms and scientific names to streamline manual curration. Give output file stem with -o.

```
./kestrel check {-t <taxa_file>} -i <input_file> -o <output_file>
```

-i, --infile=INFILE	Path to input file.
-o, --outfile=OUTFILE	Path to output csv file.
-t, --taxa=nil	Path to currated taxonomy file.

Merge
Merges search results with source file.

```
./kestrel merge {--prepend} -n <column_number>
     -r <Kestrel_output_file> -i <input_file> -o <output_file>
```

-i, --infile=INFILE	Path to input file.
-o, --outfile=OUTFILE	Path to output csv file.
--prepend	Prepend taxonomies to existing rows (appends by default).
-n, --names=NAMES	Column containing species names (integer starting from 0).
-r, --result=RESULT	Path to Kestrel search output file.