Pub2Tools

Release 1.1.1

Feb 06, 2023

Contents:

Pub2Tools is a Java command-line tool that looks through the scientific literature available in Europe PMC and constructs entry candidates for the bio.tools software registry from suitable publications. It automates a lot of the process needed for growing bio.tools, though results of the tool still need some manual curation before they are of satisfactory quality. Pub2Tools could be run at the beginning of each month to add hundreds of tools, databases and services published in bioinformatics and life sciences journals during the previous month.

CHAPTER 1

What is Pub2Tools?

Pub2Tools is a Java command-line tool that looks through the scientific literature available in Europe PMC and constructs entry candidates for the bio.tools software registry from suitable publications. It automates a lot of the process needed for growing bio.tools, though results of the tool still need some manual curation before they are of satisfactory quality. Pub2Tools could be run at the beginning of each month to add hundreds of tools, databases and services published in bioinformatics and life sciences journals during the previous month.

1.1 Overview

First, Pub2Tools gets a list of publications for the given period by *narrowing down* the entire selection with combinations of keyphrases. Next, the contents of these publications *are downloaded* and the abstract of each publication *is mined* for the potential tool name. Names are assigned confidence scores, with low confidence publications often not being suitable for bio.tools at all. In addition to the tool name, web links matching the name are extracted from the abstract and full text of a publication and *divided* to the homepage and other link attributes of bio.tools. In a *second pass*, the *content of links* and publications is also mined for *software license* and *programming language* information and phrases for the *tool description* attribute are automatically constructed. Good enough *non-existing* results are *chosen for inclusion* to bio.tools. Terms from the EDAM ontology *are added* to get the final results.

Pub2Tools can be run from start to finish with only one command (*-all*). But *setup commands* (fetching or copying the required input files) and *steps* (fetching of publications and web pages, mapping to EDAM ontology terms and applying the Pub2Tools algorithm) can also be applied individually. Execution can be resumed by restarting the last aborted step (*-resume*). Commands can be influenced by changing the default values of *parameters*. Some *examples* of running Pub2Tools are provided. One interesting example is *improving existing bio.tools entries* added through some other means than Pub2Tools.

All files of one Pub2Tools run will end up in an *output directory* chosen by the user. All prerequisite and intermediate files will be saved for reproducibility and debugging purposed. The main results files are *results.csv* (contains all possible results), *diff.csv* (contains fix suggestions to existing bio.tools content) and *to_biotools.json* (contains new entries to be imported into bio.tools). The following bio.tools attributes can be filled by Pub2Tools: name, description, homepage, function, topic, language, license, link, download, documentation, publication, credit. But not all attributes can always be filled, as shown in *Performance*, and sometimes they are filled incorrectly, so Pub2Tools results imported into bio.tools still need some fixing and manual curation. Per month, roughly 500 entries could potentially be added to bio.tools from Pub2Tools results.

1.2 Dependencies

For selecting suitable publications and downloading their content, Pub2Tools is leveraging Europe PMC, which among other things allows the inclusion of preprints.

Publications are downloaded through the PubFetcher library, that in addition to Europe PMC supports fetching publication content from other resources as fallback, for example directly from publisher web sites using the given DOI. In addition, PubFetcher provides support for downloading the content of links extracted by Pub2Tools (with support for metadata extraction from some types of links, like code repositories) and provides a database for storing all downloaded content.

Pub2Tools is also leveraging EDAMmap, for preprocessing of input free text (including the extraction of links), for downloading and loading of bio.tools content, for tf–idf support, and of course, for mapping of entries to EDAM ontology terms.

1.3 Caveats

Inevitably, there will be false positives and false negatives, both at entry level (some suggested tools are not actual tools and some actual tools are missed by Pub2Tools) and at individual attribute level. Generally, if we try to decrease the number of FN entries, the number of FPs also tends to increase. Currently, Pub2Tools has been tuned to not have too many FPs, to not discourage curators into looking at all entries in the results. Some FNs are rather hopeless: quite obviously, unpublished tools can't be found by Pub2Tools, but in addition, there is the limitation that the tool name must be mentioned somewhere in the publication title or abstract.

For slightly better results, before a bigger run of Pub2Tools, it could be beneficial to test if PubFetcher scraping rules are still up to date. Also, publisher web sites have to be consulted sometimes, so it could be beneficial to run Pub2Tools in a network with good access to journal articles.

Pub2Tools assigns a score for each result entry and orders the results based on this score. However, this score does not describe how "good" or high impact the tool itself is, but rather how confidently the tool name was extracted. A higher score is obtained if the name of the tool is unique, put to the start of the publication title, surrounded by certain keywords (like "called" or "freely") in the abstract and matches a URL in the abstract (but also in the publication full text).

1.4 Install

Installation instructions can be found in the project's GitHub repo at INSTALL.

1.5 Quickstart

This will generate results to the directory `output` for publications added to Europe PMC on the 23rd of August 2019:

```
$ java -jar path/to/pub2tools-<version>.jar -all output \
--edam http://edamontology.org/EDAM.owl \
--idf https://github.com/edamontology/edammap/raw/master/doc/biotools.idf \
--idf-stemmed https://github.com/edamontology/edammap/raw/master/doc/biotools.stemmed.
↪idf \
--day 2019-08-23
```

If this quick example worked, then for the next incarnations of Pub2Tools, the `EDAM.owl` and `.idf` files could be downloaded to local disk and the corresponding local paths used in the command instead of the URLs, and `--month`

`2019-08` could be used instead of `--day 2019-08-23` to fetch results for an entire month. Explanations for the columns and attributes of the results files can be found in the documentation at *results.csv columns*, *diff.csv columns* and *to_biotools.json attributes*.

1.6 Repo

Pub2Tools is hosted at https://github.com/bio-tools/pub2tools.

1.7 Support

Should you need help installing or using Pub2Tools, please get in touch with Erik Jaaniso (the lead developer) directly via the tracker.

1.8 License

CHAPTER 2

Usage manual

After Pub2Tools is installed, it can be executed on the command line by running the `java` command (from a Java Runtime Environment (JRE) capable of running at least version 8 of Java), while giving the compiled Pub2Tools .jar file as argument. For example, executing Pub2Tools with the argument `-h` or `--help` outputs a list of all possible parameters and commands:

```
$ java -jar path/to/pub2tools-<version>.jar --help
```

Running Pub2Tools consists of running it multiple times with different *Setup commands* that copy or fetch files required as prerequisites for the *Steps* commands, which, after being all run, will generate the end results. Commands of Pub2Tools begin with one dash (`-`) and *parameters* giving required arguments to or influencing the commands begin with two dashes (`--`). All commands must be followed by the *output directory* path where all files of a Pub2Tools run will end up.

2.1 Setup commands

Setup commands can be run in any order, and also multiple times – previous files will just be overwritten. Setup commands must be run such that all files required for the *Steps* are present in the *output directory*: *EDAM.owl*, *tf.idf*, *tf.stemmed.idf*, *biotools.json*, *pub.txt* and *db.db*.

Note: Once any *Steps* has completed successfully, no Setup commands can be run anymore and only Steps commands are allowed to be run to finalise the current Pub2Tools run.

2.1.1 -copy-edam

Copy the EDAM ontology file in OWL format given with the `--edam` parameter to *EDAM.owl* in the given output directory. The EDAM ontology is used in the last *-map* step to add EDAM annotations to the results.

The given OWL file can be a path to a file in the local file system or a web link, in which case the file will be downloaded from the link to the given output directory. Fetching of the link can be influenced by the –timeout and –user-agent parameters.

Note: In the same way, either a local file or a web resource can be given as the source file for all following `-copy` commands.

Examples copying the EDAM ontology file to the directory `results`:

```
$ java -jar path/to/pub2tools-<version>.jar -copy-edam results --edam path/to/EDAM.owl
$ java -jar path/to/pub2tools-<version>.jar -copy-edam results --edam http://
↪edamontology.org/EDAM.owl
```

2.1.2 -copy-idf

Copy two IDF files (one where words are stemmed and another where they are not) given with the `--idf` and `-idf-stemmed` parameters to *tf.idf* and *tf.stemmed.idf* in the given output directory. tf–idf weighting is used in multiple parts of Pub2Tools: the unstemmed version (`tf.idf`) is used in *-pass1* and *-pass2* and in the *-map* step either `tf.idf` or `tf.stemmed.idf` is used, depending on the used parameters (using stemming is the default).

Pre-generated IDF files are provided in the EDAMmap repo: biotools.idf and biotools.stemmed.idf. However, these files can also be generated from scratch using EDAMmap: more info in the IDF section of the EDAMmap manual.

Example copying the (either downloaded or generated) IDF files from their location on local disk to the `results` directory:

```
$ java -jar path/to/pub2tools-<version>.jar -copy-idf results --idf path/to/tf.idf --
↪idf-stemmed path/to/tf.stemmed.idf
```

2.1.3 -get-biotools

Fetch the entire current bio.tools content using the bio.tools API to the file *biotools.json* in the given output directory. This file containing existing bio.tools content is used in the *-pass2* step to see which results are already present in bio.tools.

Calls the same code as the EDAMmap-Util command `-biotools-full`.

Example fetching bio.tools content to the `results` directory:

```
$ java -jar path/to/pub2tools-<version>.jar -get-biotools results
```

2.1.4 -copy-biotools

Copy the file containing bio.tools content in JSON format given with the `--biotools` parameter to the file *biotools.json* in the given output directory. This `-copy` command can be used instead of *-get-biotools* to re-use a biotools.json file downloaded with `-get-biotools` as part of some previous Pub2Tools run or to use some alternative JSON file of bio.tools entries.

Example copying bio.tools content to the `results` directory:

```
$ java -jar path/to/pub2tools-<version>.jar -copy-biotools results --biotools path/to/
↪biotools.json
```

2.1.5 -select-pub

Fetch publication IDs of journal articles from the given period that are potentially suitable for bio.tools to the file *pub.txt* in the given output directory. The resulting file is used as input to the *-fetch-pub* step that will download the content of these publications forming the basis of the search for new tools and services to add to bio.tools. Only articles matching certain (changeable) criteria are selected, as otherwise the number of publications to download for the given period would be too large – due to this filtering the number of publications is reduced by around 50 times (with default options).

The granularity of the selectable period is one day and the range can be specified with the parameters `--from` and `--to`. As argument to these parameters, an ISO-8601 date must be given, e.g. `2019-08-23`. Instead of `--from` and `--to` the parameters `--month` or `--day` can be used. The parameter `--month` allows to specify an exact concrete month as the period (e.g. `2019-08`), so that the number of days in a month doesn't have to be known to cover any whole month. The parameter `--day` allows to specify just one whole day as the period (e.g. `2019-08-23`).

Fetching the publication IDs works by sending large query strings to the Europe PMC API and extracting the PMID, PMCID and DOI from the returned results. The query string consists of the date range, of the content source, of (`OR`-ed together) phrases that must be present in the abstract to try to restrict the output to articles about tools, of a potential custom search string to restrict the output further to the desired theme and of (`AND`-ed together) phrases that must not appear in the publication abstract or title to try to remove some false positives:

- The date range is specified with `--from` and `--to` or `--month` or `--day`, as explained above. The search field filled with the specified date is "CREATION_DATE", which is the first date of entry of the publication to Europe PMC. This is not necessarily equal to the (print or electronic) publication date of the journal article, as a publication can be added to Europe PMC some time after it has been published, but also ahead of publication time. The search field "CREATION_DATE" is used instead of publication date to try to ensure that the set of publications returned for some date range remains the same in different points in time. For example, if all publications of August are queried at some date in September, we would want to get more or less the same results also at some query date in October. If the article publication date was used as the search field, then maybe some articles published in August were added to Europe PMC in the meantime, meaning that the query made in October would return more results and the query made in September would miss those newly added publications. Using "CREATION_DATE" enables us to do the query of publications added to Europe PMC in August only once and not bother with that date range anymore in later queries.
- Europe PMC has content from different sources (Sources of content in Europe PMC). Pub2Tools searches from "MED" (PubMed/MEDLINE NLM) and "PMC" (PubMed Central). As we are only interested in the PMID and PMCID, then we request a minimal amount of information in the results to save bandwidth (`resultType=idlist`).

  But Pub2Tools also searches from the source "PPR" (Preprints). The inclusion of preprints in Europe PMC is a nice feature that enables Pub2Tools to easily extend the search of tools to publications in services like bioRxiv and F1000Research. In case of preprints we usually can only get a DOI and, as the minimal results do not contain it, we execute the query for publication IDs from preprints separately (`resultType=lite`).
- Phrases that must appear in a publication abstract are divided into categories and by combining these categories we get the final necessary requirement a publication abstract must meet to be included in the selection. Files corresponding to these categories are the following:
  - excellent, e.g. "github", "implemented in r", "freely available"
  - good, e.g. "available for academic", "sequence annotation", "unix"
  - mediocre1, e.g. "our tool", "paired-end", "ontology"
  - mediocre2, e.g. "computationally", "high-throughput", "shiny"
  - http, e.g. "https", "index.html"
  - tool_good, e.g. "server", "plugin"

  - tool, e.g. “tool”, “pipeline”, “repository”

Out of these, only phrases from `excellent` are sufficient on their own to meet the inclusion requirement. Phrases from other categories must be combined, e.g. an abstract matching one phrase from `good` and another from `http` will also meet the requirement. This can be directly encoded as an Europe PMC query by `AND`-ing together the `OR`-ed phrase of `good` with the `OR`-ed phrase of `http`. However, some combinations can't be expressed as Europe PMC queries, for example one phrase from `good`, another from `tool` and a third, but different one also from `tool`. In that case, results for all phrases of `tool` must be fetched from Europe PMC one by one and programmatically combined in Pub2Tools. In total, the following combinations are done:

  - `excellent`
  - `good` + `http`
  - `good` + `tool_good`
  - `mediocre1` + `http` + `tool`
  - `mediocre2` + `http` + `tool`
  - `mediocre1` + `tool_good` + `tool`
  - `mediocre2` + `tool_good` + `tool`
  - `http` + `tool_good`
  - `good` + `tool` + `tool`
  - `mediocre` + `tool` + `tool` + `tool`
  - `http` + `tool` + `tool`
  - `tool_good` + `tool_good`
  - `tool_good` + `tool` + `tool`

Note: The category `mediocre` is split into two in the implementation simply because otherwise query strings sent to Europe PMC would get too long for it to handle.

Note: It seems that common words (stop words maybe) are filtered out, e.g. `ABSTRACT:"available as web"` and `ABSTRACT:"available at web"` give the same results (however `ABSTRACT:"available web"` gives different results, so some stop word must be present in-between).

This restricting by phrase combinations is done to considerably narrow down the returned number of publication IDs so that the amount of publication content to be downloaded would be reasonable. It also reduces potential false positives further down the line, but inevitably, some good articles about tools also get discarded in the process. To not do this restricting, the parameter `--disable-tool-restriction` can be supplied. However, doing this would significantly increase the number of results (mostly in the form of false positives), so `--disable-tool-restriction` should really be used in conjuction with `--custom-restriction`.

- A custom search string can be specified after the parameter `--custom-restriction` to further restrict the results, for example to some desired custom theme. How to construct such a search string, what fields can be searched and how to combine them can be seen in Search syntax reference of Europe PMC. For example, something like `--custom-restriction '"COVID-19" OR "SARS-CoV-2" OR ABSTRACT:"Coronavirus"'` could be used to restrict the output of Pub2Tools to tools related to the COVID-19 pandemic (in reality, the search string should be made a bit more elaborate). Constructed search strings can be tested using the search box at https://europepmc.org/. If the search string is specific enough and looking through every hit is important, then other restrictions can potentially be disabled with `--disable-tool-restriction` and potentially also `--disable-exclusions`.

- As the last part of the query string sent to Europe PMC, phrases that must not appear in the publication abstract or title are specified to remove some systematic false positives. These help to exclude a few publications that are otherwise selected, but that are actually not about a tool or service (mostly, publications about a medical trial and review articles are excluded this way). The exclusion phrases are specified in the files not_abstract.txt (e.g. "trial registration", "http://clinicaltrials.gov") and not_title.txt (e.g. "systematic review", "controlled trial"). To not do such exclusions, the parameter `--disable-exclusions` can be supplied (this would mostly just have the effect of introducing a small number of additional FPs).

Some journals have articles suitable for bio.tools more often than some other journals. As the selection of publications with phrases that must appear in the abstract is not perfect and sometimes excludes good articles, it makes sense to not use this mechanism for some high relevance journals and instead download all publications of the given period from these journals. If the number of such journals is not too high, then this does not significantly increase the total number of publications that must be downloaded. The list of such high priority journals is specified in the file journal.txt. Phrase exclusion with `not_abstract.txt` and `not_title.txt` is still done (unless `--disable-exclusions` is specified) and additional restrictions from `--custom-restriction` will also apply. Separate selection from these journals is not done if the parameter `--disable-tool-restriction` is specified.

Two equivalent examples fetching all publication IDs for the month of August 2019 to the directory `results`:

```
$ java -jar path/to/pub2tools-<version>.jar -select-pub results --from 2019-08-01 --
↪to 2019-08-31
$ java -jar path/to/pub2tools-<version>.jar -select-pub results --month 2019-08
```

Example selecting publication IDs from publications added to Europe PMC on the 23rd of August 2019:

```
$ java -jar path/to/pub2tools-<version>.jar -select-pub results --day 2019-08-23
```

Example selecting publication IDs of COVID-19 related articles for the year 2020 (normally, selecting large time spans can get too slow because of large numbers of combinations to be done with too many returned IDs, here, `--custom-restriction` is restricting the output enough):

```
$ java -jar path/to/pub2tools-<version>.jar -select-pub results --from 2020-01-01 --
↪to 2020-12-31 --custom-restriction '"2019-nCoV" OR "2019nCoV" OR "COVID-19" OR
↪"SARS-CoV-2" OR "COVID19" OR "COVID" OR "SARS-nCoV" OR ("wuhan" AND "coronavirus")␣
↪OR "Coronavirus" OR "Corona virus" OR "corona-virus" OR "corona viruses" OR
↪"coronaviruses" OR "SARS-CoV" OR "Orthocoronavirinae" OR "MERS-CoV" OR "Severe␣
↪Acute Respiratory Syndrome" OR "Middle East Respiratory Syndrome" OR ("SARS" AND
↪"virus") OR "soluble ACE2" OR ("ACE2" AND "virus") OR ("ARDS" AND "virus") or (
↪"angiotensin-converting enzyme 2" AND "virus")'
```

2.1.6 -copy-pub

Copy the file containing publication IDs to download with *-fetch-pub* from the path given with the `--pub` parameter to the file *pub.txt* in the given output directory. This `-copy` command can be used instead of *-select-pub* in order to use publication IDs got through some different means (for example, a list of publication IDs could be manually created).

Example copying the file containing publication IDs to the `results` directory:

```
$ java -jar path/to/pub2tools-<version>.jar -copy-pub results --pub path/to/pub.txt
```

2.1.7 -init-db

Initialise an empty PubFetcher database to the file *db.db* in the given output directory. The database is used to store the contents of the publications fetched with *-fetch-pub* and webpages and docs fetched with *-fetch-web*. The database is read for getting the contents of publications in *-pass1*, for the contents of webpages and docs in *-pass2* and for all content in *-map*.

Note: In contrast to other Setup commands, `-init-db` will not automatically overwrite an existing file (as the filling of an existing database file might have taken a lot of resources), so `db.db` must be explicitly removed by the user if `-init-db` is to be run a second time.

Calls the same code as the PubFetcher CLI command -db-init.

Example initialising an empty database file to the `results` directory:

```
$ java -jar path/to/pub2tools-<version>.jar -init-db results
```

2.1.8 -copy-db

Copy the PubFetcher database given with the `--db` parameter to the file *db.db* in the given output directory. This can be used instead of *-init-db* in order to use a database full of publications and web pages got through some other means.

Example copying an existing database to the `results` directory:

```
$ java -jar path/to/pub2tools-<version>.jar -copy-db results --db path/to/db.db
```

2.2 Steps

Once setup is done, steps must be run in the given order: *-fetch-pub*, *-pass1*, *-fetch-web*, *-pass2* and *-map*. A re-run, starting from any step, is also possible – previous results will be overwritten. And if there is confidence in the set of publications and web pages not changing, then *-fetch-pub* and *-fetch-web* can be skipped, if they have been run at least once. Although running also *-fetch-pub* and *-fetch-web* a second time might be beneficial in that some previously inaccessible or slow web resources might now be online. After a step successfully concludes, the next step to be run is written to *step.txt*. Once all steps have completed successfully, the files *results.csv*, *diff.csv* and *to_biotools.json* will be present in the given *output directory*.

2.2.1 -fetch-pub

This will fetch publications for publication IDs given in *pub.txt* to the database file *db.db* in the given output directory. Fetching is done like in the -db-fetch-end method of PubFetcher. Fetching behaviour can be influenced by the Fetching parameters and by `--fetcher-threads` that sets how many threads to use for parallel fetching (default is 8).

For best results, before a major run PubFetcher scraping rules could be tested with the PubFetcher CLI command `-test-site`, especially if this hasn't been done in a while. Also for better results, `-fetch-pub` could potentially be run multiple times, spaced out by a few days, as some web pages might have been temporarily inaccessible the first time. A re-run is quicker as fetching is not retried for resources that were fetched to final state the first time. And also for better results, sometimes full texts of publications are downloaded directly from publisher sites, thus using Pub2Tools in a network with better access to those is beneficiary.

Example of running the step with some non-default parameter values:

```
$ java -jar path/to/pub2tools-<version>.jar -fetch-pub results --timeout 30000 --
↪journalsYaml fixes.yaml --fetcher-threads 16
```

2.2.2 -pass1

The first pass of the Pub2Tools algorithm will load all publications from *db.db* corresponding to the publication IDs in *pub.txt* and iterate over these publications trying to find a **name** for the tool or service each publication is potentially about, assign a goodness **score** to the name suggestion and try to find web **links** of the tool from the publication abstract and full text. The unstemmed *tf.idf* is also read, as tf–idf weighting is used as part of the scoring and link matching. Results are output to *pass1.json* (for input to the second pass *-pass2*) and matched links to *web.txt* and *doc.txt* (so that *-fetch-web* can download their contents).

Publications whose abstract length is larger than 5000 or full text length is larger than 200000 are discarded altogether. Then, text from publications needs to be preprocessed – support for this comes from EDAMmap. Input is tokenised and processed, for example everything is converted to lowercase and stop words are removed. Processing is good for doing comparisons etc, however tokens closer to the original form (e.g. preserving the capitalisation) are also kept, as this is what we might want to output to the user. Code to divide the input into sentences and to extract web links has also been implemented in EDAMmap and is used here. This implementation might not be perfect, but it has enabled devising regexes and hacks dealing with quirks and mistakes specific to the input got from publications. The removal of stop words and some other preprocessing (except `--stemming`) can be influenced by the Preprocessing parameters.

Then, the process of looking at all possible phrases in the publication title and abstract as potential names of a tool or service begins. The goodness scores of the phrases are calculated and modified along the way:

- First, words in the title and abstract are scored according to tf–idf weighting, using the `tf.idf` file generated from bio.tools content. A unique word (according to bio.tools content) appearing once in the abstract will get a score of 1. The more common the word is, the lower the score according to a formula. If the word occurs more than once in the title and abstract, then the score will be higher. Short phrases (many words as a tool name) are also calculated scores for, using the scores of their constituent words.
- Quite often, the tool name is present in a publication title as "Tool name: foo bar", "Tool name - a foo bar", etc. Extracting the phrase before ": ", " - ", etc, and removing some common words like "database", "software", "version", "update", etc, from that phrase would result in a phrase (the *tool_title*) that we have more confidence in being the tool name. Thus, we increase the score of that phrase by multiplying its score from the last step with a constant (or initialise it to a constant if the extracted phrase is a new combination). The `tool_title` could also be an acronym with the expanded name occurring somewhere in the abstract. Fittingly, matching acronyms to their expanded forms is also supported (here and in the next steps).
- In a publication abstract about a tool, certain words tend to occur more often just before or after the tool's name than they occur elsewhere. So, if a candidate phrase has one such word before or after it, the probability that the phrase is a tool name is higher and we can increase its score. The list of such words that often occur just before or after (or one step away) from a tool name was bootstrapped by tentatively setting as tool name the *tool_title* (where available). These bootstrapped words were divided into tiers based on how much they preferably occur around the `tool_title`, thus how much they should increase the score. For example, the best words to occur before a tool name are in before_tier1.txt (e.g. "called", "named") and after a tool name in after_tier1.txt (e.g. "freely", "outperforms"). Words raising the score less are in before_tier2.txt and after_tier2.txt, before_tier3.txt and after_tier3.txt. Now, having these word lists, we can iterate through each candidate phrase in the title and abstract and raise the score by some amount depending on the tier (but up to a limit) each time when a "before" or "after" word is found to be in the neighbourhood.
- If an abstract contains web links, we can be somewhat more certain that the publication is about a software tool or service, as in such publications links to the tool are often put in the abstract. However, such links can point to other things as well, for example to some resource used in the publication. So what we would like to do, is to match these links to phrases in the abstract and increase the score of candidate phrases that have matching

links. In addition to matching links in the abstract, it also makes sense to match the candidate phrases from the abstract to links in the full text of the publication (while having a smaller matching score in that case), as often the homepage of the tool is not put into the abstract or additional links can appear in the full text (the repository of the tool, some documentation links, etc). The matching of links to phrases increases the score of some phrases and thus helps in finding the most likely tool name, but in addition, once the name has been chosen, we can possibly suggest a homepage, documentation links, etc, (done in *-pass2*) based on the links attached to the name.

The matching of links is done by extracting the part of the link URL string that is most likely a tool or service name and matching it in various forms (including in acronym form) to the candidate phrase. The part extracted from the URL string is either a path part, or if there is no path or all path parts are too unlikely, then it is extracted from the domain name. Choosing the correct path part is done from right to left with the unlikeliness of being the tool name decided mainly by tf–idf weighting. If the name has to be extracted from the domain name, then the lowest level domain name part is chosen, unless it matches some hardcoded patterns or any of the words in host_ignore.txt (in which case, the link can't be matched to any phrases at all).

In some cases, the tool or service name can correctly be extracted from the link, however it doesn't match any phrases in the publication title or abstract simply because the tool name is not mentioned there. To also catch and potentially include such publications, such orphaned link parts are added to the pool of candidate phrases (*from_abstract_link* of such name suggestions is set to `true`). In some other cases, the matching of links fails for some other reason or the extraction of the link part fails to work correctly, so as a backup mechanism, candidate phrases are also matched to any part of a link URL (but in case of a match, the score of the phrase is not increased); and if this also fails, then in case a variation of the word "available" appears in the same sentence as the unmatched link, that link will be attached to the phrase (again, the score of the phrase is not increased), unless it appears to be a link to a dataset repository.

Once the final score has been calculated, candidate phrases are ordered by it and the top one suggested as the tool or service name. Up to 4 more candidates can be output, if their scores are not too low compared to the top one. This usually means that for publications where the confidence in the top choice is not very high, other options besides the top one will also appear.

The publications themselves can also be ordered based on the scores of their top choices and possibly a threshold could be drawn somewhere, below which we would say that the publication is not about a tool or service (however, such *final decision* will be done in the end of the second pass by taking some additional aspects into account).

Note: A higher score does not mean a "better", higher impact, etc tool. It just mean that Pub2Tools is more confident about the correctness of the extracted tool name.

Note: One publication can possibly be about more than one tool. Currently we only detect this when the names of such tools are in the title and separated by "and" or "&" – in such case we split the publication into independent results for each tool.

The final output of the first pass of Pub2Tools needs some cleaning:

- For example, there are often problems with web links: sometimes links are "glued" together and should be broken into two separate links, sometimes there seems to be garbage at the end of a link, sometimes the schema protocol string in front of the URL is truncated, etc. We can fix some such mistakes by guesswork or sometimes a problematic link in the abstract has a correct version in the full text or vice versa. After fixing the links, we also keep the unfixed versions, because they might have been correct after all (this will be known after trying to resolve the links).
- Mistakes in the source material can cause other output to be invalid also. For example, the publication DOIs sometimes contain garbage in them that causes them to be discarded.

- Even if the output seems to be correct, it has to be valid according to biotoolsSchema, and this can cause further modifications to be made. For example, some attribute values might need to be truncated because of maximum length requirements. Or, according to biotoolsSchema, the extracted tool name can only contain Latin letters, numbers and a few punctuation symbols and thus, invalid characters are either replaced (accents, Greek letters, etc) or discarded altogether.

In the end, results are written to *pass1.json* for further processing by *-pass2*. Results contain the publication IDs and other information about the publication, like the title, possible name extracted from the title (*tool_title*), sentences from the abstract, journal title, publication date, citations count, corresponding authors, but also the suggested tool name (or names in case of multiple suggestions) along with the suggestion's score and links from the abstract and full text matching the name. All matched *links are divided* to documentation and other links (based on the URL string alone) and written to *doc.txt* and *web.txt* for fetching by the next step.

Example of running the step:

```
$ java -jar path/to/pub2tools-<version>.jar -pass1 results
```

2.2.3 -fetch-web

This will fetch webpages and docs for URLs given in *web.txt* and *doc.txt* to the database file *db.db* in the given output directory. Fetching is done like in the -db-fetch-end method of PubFetcher. Fetching behaviour can be influenced by the Fetching parameters and by `--fetcher-threads` that sets how many threads to use for parallel fetching (default is 8).

For best results, before a major run PubFetcher scraping rules could be tested with the PubFetcher CLI command `-test-webpage`, especially if this hasn't been done in a while. Also for better results, `-fetch-web` could potentially be run multiple times, spaced out by a few days, as some web pages might have been temporarily inaccessible the first time. A re-run is quicker as fetching is not retried for resources that were fetched to final state the first time.

Example of running the step with some non-default parameter values:

```
$ java -jar path/to/pub2tools-<version>.jar -fetch-web results --timeout 30000 --
↪webpagesYaml fixes.yaml --fetcher-threads 16
```

2.2.4 -pass2

The second pass of the Pub2Tools algorithm will load all results of the first pass from *pass1.json* and while iterating over these results it will: reassess and reorder entries with lower scores by calculating a **second score**; **merge entries** if they are determined to be about the same tool; look into *biotools.json* to see if an entry is **already present** in bio.tools; assign **types to all the links** and decide which one of them is the **homepage**. The unstemmed *tf.idf* is also read, as tf–idf weighting is used as one part of calculating the second score, and *db.db* is read to get the license, language and description candidate phrases from webpages and docs and to check if webpages and docs are broken and if the final URL (after redirections) is different than the initial one. Final results of all entries are output to *results.csv* along with intermediate results, new entries determined to be suitable for entry into bio.tools are output to *new.json* with the output adhering to biotoolsSchema and differences between the content of entries determined to already exist in bio.tools and the corresponding content in bio.tools are output to *diff.csv*.

Calculate score2 A second score is calculated for entries whose first score (calculated in *-pass1*) is below 1000. This has as goal elevating entries that are quite likely about a tool but that got a low score in the first pass (showing trouble in being confident in the extracted tool name). Up to 5 tool names are suggested for an entry after the first pass with the top name being suggested as the correct one, however this top name choice can potentially change while calculating score2 for all tool name suggestions of an entry. For calculating score2, first it is set equal to the first score and then:

1. If a suggestion has matching non-broken links in the publication abstract, then increase its score2 (matching non-broken links in the fulltext also increase score2, but less). It's possible, that the top name suggestion changes after this step, as the current top name might not have matching links, but some lesser choice might.
2. If a suggestion matches the tool name that can be extracted from the publication title (*tool_title*), then increase its score2 depending on how good the match is and how many words the name contains (less is better). Again, the suggestions can get reordered and the top name suggestion change.
3. Next, increase score2 of a suggestion based on the capitalisation of the name. A mix of lower- and uppercase letters gets the highest increase and all lowercase letters the smallest, and if the name consists of many words, then the score increase is lowered. The score2 of other suggestions besides the current top one is only increased if it was already increased by any of the two previous methods.
4. Increase score2 of a suggestion based on the average uniqueness of the words making up the tool name (calculated based on the input *tf.idf* file). The score2 of other suggestions besides the current top one is only increased if it was already increased by any of the first two methods.

Determine confidence If the first score is at least 1000 or *score2* is more than 3000, then confidence is "high". If score2 is more than 1750 and less or equal to 3000, then confidence is "medium". If score2 is at least 1072.1 and less than or equal to 1750, then confidence is "low". If score2 is less than 1072.1, then confidence is "very low".

Note: The confidence value is more like the confidence in the correctness of the extracted tool name. Whether an entry should be considered to be about a tool and thus eligible for entry to bio.tools is not based solely on this confidence and the *final decision* of inclusion is done later.

Merge same suggestions In the first pass (*-pass1*) a few *publications were split*, as sometimes a publication can be about more than one tool. Conversely, different publications can also be about the same tool, so here we try to merge different entries into one entry for these kinds of publications. This merging is done, if the top name suggestions of the entries are exactly equal and the confidences are not "very low". Entries with a "very low" *confidence* are not merged and instead connected through the *same_suggestion* field.

Note: Entries with a "very low" confidence can in some occasions also be included in *new.json*, thus the tool name is not a unique identifier there.

Find existing bio.tools entries If an entry is found to be existing in bio.tools, then it should not be suggested as a new addition in *new.json*. However differences with the current bio.tools content are highlighted in *diff.csv*.

Note: What is meant under the current bio.tools content is not the content of bio.tools at the exact time of running `-pass2`, but the content in the supplied *biotools.json* file.

An entry is determined to be **existing in bio.tools** in the following cases:

- Some publication IDs of the new entry are matching some publication IDs of an entry in bio.tools.
- The name of the entry is equal or matching (ignoring version, capitalisation, symbols, etc) a name or ID of an entry in bio.tools and some link from that entry is matching (ignoring lowest subdomain and last path) any link from that bio.tools entry. As additional requirement in this case, the *confidence* must not be "very low" and the *final decision* about inclusion must be positive.
- The name of the entry is matching a name or ID of an entry in bio.tools and also matching a *credit* (through the name, ORCID iD or e-mail) with that bio.tools entry. The *confidence* must not be "very low" and the *final decision* about inclusion must be positive.

Divide links The web links extracted in *-pass1* from publication abstracts and full texts and matching with the tool name suggestion are divided into bio.tools link groups and assigned a type. One of these links is chosen to be the tool homepage and broken links are removed.

The link groups of bio.tools are link, download and documentation. Further inside the group, each link is assigned a type, e.g. "Mailing list", "Source code" or "Installation instructions". In Pub2Tools, the division of links and assignment of type is done solely based on matching regular expressions to the link URL string. For example, a URL ending with `".jar"` would be a `download` with type "Binaries", matching of `"(?i)(^|[^\\p{L}])faqs?([^\\p{L}]|$)"` would be `documentation` with type "FAQ" and matching of the host `"github.com"` would be a `link` with type "Repository". Note, that some other link types might also have the host `"github.com"`, for example the GitHub tab "Issues" is put under `link` type "Issue tracker", the GitHub tab "Wiki" is put under `documentation` type "User manual" and "Releases" is put under `download` type "Software package" – so these options would have to be explored before `link` type "Repository". Links whose URL can't be matched by any of the rules will end up under `link` type "Other".

After division, links determined to be broken (i.e. that were not successfully resolved or returned a non-successful HTTP status code) are removed from `link`, `download` and `documentation`.

In the end, one of the links is chosen as the bio.tools homepage. First, only links in the abstract are considered with the following priority:

1. `link` "Other"
2. `link` "Repository"
3. `documentation` "General"
4. other `documentation`
5. any non-broken link whose URL does not end with an extension suggesting it is a downloadable file

If no suitable homepages are found this way, then links in the fulltext are considered following the same logic. If a homepage is still not found, then broken links in the abstract or fulltext are also allowed to be the homepage (and the status is set to *"broken"*, if such a homepage is found). And if there are still no suitable homepages, then a link to the publication (in PubMed, in PubMed Central or a DOI link) is set as the homepage (and the status is set to *"missing"*).

Description As the bio.tools description attribute needs to be filled, then from all fetched and gathered content some candidate phrases are generated, that the curator can choose from or combine into the final description. The more difficult task of automatic text summarisation has not been undertaken, thus the `description` is an output of Pub2Tools that definitely needs further manual curation (along with EDAM terms output in *-map*).

As the length of the `description` is limited to 1000 characters, then phrases from different sources need to be prioritised and only some can be chosen for consideration. First, the publication title is output as a potential description (with the potential tool name, i.e. *tool_title*, removed from it).

Then, a suitable description phrase is looked for in web page titles. From these titles, any irrelevant content and the tool name are potentially removed with simple heuristics and a minimum length is required for such modified titles to be considered. The next priority goes to the first few sentences of a web page that are long enough. But one or two sentence (one short and one longer) that contain the tool name are also looked for in any case, with the priority of such sentences depending on how far from the top of the page they are. Also, the priority of sentences from some web page are a bit higher if scraping rules for that web page exist in PubFetcher. In case of equal priority, phrases from the homepage are preferred (then from certain `link` types and then from `documentation`). Menus, headers and other non-main content of web pages are mostly ignored due to PubFetcher's ability to filter out such content. Deduplication of the final suggested phrases is also attempted.

If the search for potential phrases from web pages does not yield any results, then the description candidate phrases after the publication title will be from the publication abstract.

In addition to the Pub2Tools results themselves, we might want to communicate to the curator things that should potentially be checked or kept in mind when curating the entry (for example, that the homepage is potentially

"broken" or that there is a slight chance that the entry is already *existing in bio.tools*). As there are no nice and non-hidden places for such messages in the output meant for bio.tools, then the `description` attribute is abused for such purpose. The messages to the curator, if any, will be appended to the description candidate phrases, are separated by `"\n\n"`, prefixed with `"|||"` and written in all caps. The space reserved for the messages is up to half of `description` (500 characters), any non-fitting discarded messages will be logged. The list of possible messages can be seen in the *output* documentation at *description*.

License License information for the bio.tools license attribute is looked for in the `homepage` and in the `link`, `download`, `documentation` links (in the license field provided by PubFetcher) and also in the publication abstract. The most frequently encountered license is chosen as the suggested license.

An encountered license string must be mapped to the SPDX inspired enumeration used in bio.tools (see license.txt). Difficulties in doing so include the fact, that many licenses have versions (which can be specified in different ways, like "GPL-3", "GPL(>= 3)" or "GPLv3", or not specified at all) and even the license name can be specified differently (as an acronym or fully spelled out or something in-between or there are just different ways to mean the same license). In free text, like the publication abstract, some license strings from the enumeration can sometimes match words in the text that are actually not about a license, so require the presence of `"Licen[sc]"` in the immediate neighbourhood of such matches to avoid false positives.

Language Language information for the bio.tools language attribute is looked for in the `homepage` and in the `link`, `download`, `documentation` links (in the language field provided by PubFetcher) and also in the publication abstract. All encountered languages (with duplicates removed) are chosen as the suggested languages.

An encountered language string must be mapped to the programming language enumeration used in bio.tools (see language.txt). Most language strings are rather unique, so these can relatively safely be matched by carefully extracting the characters from the target text. However, some languages (like "C", "R", "Scheme") can easily be mistaken for other things, so the presence of a keyword (like "implemented", "software", "language", full list in language_keywords.txt) in the immediate neighbourhood is required in such cases. Somewhat conversely, some words can automatically infer a language (like "bioconductor" -> "R", "django" -> "Python").

Credit Currently, information to add to the bio.tools credit attribute comes only from one place: the corresponding authors of publications. Corresponding authors can be extracted from articles in PubMed Central, i.e. for publications that have a PMCID, or names and e-mails can also be extracted from many journal web pages of articles, i.e. from web pages got after resolving the DOIs. The task of extracting corresponding author information is done by code from PubFetcher and stored in the publication field correspAuthor. We can possibly get the name, ORCID iD, e-mail, phone and web page of a corresponding author with PubFetcher. The only thing we can additionally do here, is merge potential duplicate authors coming from different sources (and there are some things to keep in mind when merging, for example the name of the same person can be written with or without potentially abbreviated middle names, academic titles, accents, etc).

Final decision The final decision whether an entry is suggested for entry to bio.tools (when it's determined to not already *exist in bio.tools*) is not based solely on *confidence*, but there are some further considerations. First, if confidence is "high", then it is suggested for entry. For any lower confidence, it is suggested for entry if the first score (not *score2*) is high enough (the threshold depending on the exact confidence) or in case the first score is too low, then 1 or 2 (depending on score) of the following has to hold:

1. the homepage is not *"missing"*;
2. a *license* is found;
3. at least one *language* is found;
4. none of the publications have a PMID or PMCID (i.e. all publications have only a DOI).

In addition, certain homepage suggestions (like "clinicaltrials.gov") or journals (like "Systematic reviews") will also bar the entry from being suggested (even if confidence is "high"). Also, the presence of words from not_abstract.txt in the abstract or words from not_title.txt in the publication title will exclude the entry (although, publications containing these words were already excluded by *-select-pub*, if the normal workflow was used).

All entries (irrespective of the *final decision* on whether the entry should be added to bio.tools) are output to *results.csv* along with all possible data (explained in *results.csv columns*). *Merged entries* are output as one entry, with values of the constituent entries separated by " | " in the field values.

If an entry is determined to be about a tool already *existing in bio.tools*, then it is not suggested for entry to bio.tools. However, if there are any differences between values of the entry and values of the corresponding entry in bio.tools or if bio.tools seems to be missing information, then these differences and potential extra information are added to *diff.csv* (with a detailed explanation of the values in *diff.csv columns*).

Note: Sometimes, an entry seems to be existing in bio.tools, but the evidence for existence is not enough (and more often than not misleading) – in that case the new entry is still suggested for entry to bio.tools, but information about the potentially related existing entry in bio.tools is output as a message in the *description*.

All entries for which the *final decision* is positive and that are determined to not be *existing in bio.tools* are added to *new.json* in biotoolsSchema compatible JSON. Attributes that can be filled are:

- `name`, as extracted by *-pass1*
- `publication`, originally selected in *-select-pub*
- `description`, `license`, `language`, `credit`, constructed here in the second pass (explained in *description*, *license*, *language*, *credit*)
- `homepage`, `link`, `download`, `documentation`, as described in *divide links*

Example of running the step:

```
$ java -jar path/to/pub2tools-<version>.jar -pass2 results
```

2.2.5 -map

This step will add EDAM ontology annotations using EDAMmap to the Pub2Tools results in *new.json*, outputting the annotated results to *to_biotools.json* in the given *output directory*. Additional inputs to the mapping algorithm are the EDAM ontology file *EDAM.owl*, the database file *db.db* containing the contents of publications, webpages and docs, and the IDF files *tf.idf* and *tf.stemmed.idf* (which IDF file being used depending on the supplied `--stemming` parameter, with the stemmed version being the default). As additional output, more details about the mapping results are provided in different formats and detail level in *map.txt*, *map/ directory of HTML files* and *map.json*.

The input and output file names of the mapping step are fixed and cannot be changed. However, other aspects of the mapping process can be influenced by a multitude of parameters: Preprocessing parameters, Fetching parameters and Mapping parameters. By default, the default parameter values are used, for example up to 5 terms from the "topic" and "operation" branches are output (with results from the "data" and "format" branches being omitted by default, as currently EDAMmap does not work well for "data" and "format"). In addition, the parameter `--mapper-threads` can be used to set how many threads to use for parallel mapping of entries (default is 4).

The mapping step will in essence just fill and add the operation attribute (under a new function group) and the topic attribute, containing a list of EDAM term URIs and labels, to the new bio.tools entries in *new.json*, outputting the result to *to_biotools.json*.

Annotations from the "data" and "format" branches can also be added if requested. However, in the function group of bio.tools, it has to be specified whether the data attribute and the format attribute are input or output data and format, and EDAMmap can't do that. So, if mapping results for the "data" and "format" branches are to be output, they will be output to the function note attribute as text and not to the data and format attributes.

In addition to the *description* attribute, the EDAM terms output by Pub2Tools are attributes that definitely need further manual curation. Unfortunately, EDAMmap scores of the found terms cannot be output to *to_biotools.json*, which makes it a bit harder to decide on the correctness of the suggested terms. However, suggested concepts are ordered by

score so it can be assumed, that the few last term suggestions are more probably wrong. And if needed, more detailed mapping results (including scores) can be found in *map.txt*, *map/ directory of HTML files* or *map.json*.

Example of running the step with some non-default parameter values:

```
$ java -jar path/to/pub2tools-<version>.jar -map results --stemming false --branches␣
↪topic operation data format --mapper-threads 8
```

2.2.6 -all

This command will run all *setup commands* necessary for getting the files required by the *steps* and then run all steps in order (*-fetch-pub*, *-pass1*, *-fetch-web*, *-pass2*, *-map*). So in essence, it could the sole command run to get a batch of new results from Pub2Tools.

The command has a few mandatory parameters: `--edam` to copy the EDAM ontology (as in *-copy-edam*), `--idf` and `--idf-stemmed` to copy the IDF files (as in *-copy-idf*) and `--from`/`--to` or `--month` or `--day` to specify a date range for fetching publication IDs (as in *-select-pub*). The last date range parameters can actually be replaced with `--pub`, that is, instead of fetching new publication IDs, a file containing publication IDs can be copied (as in *-copy-pub*).

In addition, the parameter `--biotools` can be specified to copy a JSON file containing the entire content of bio.tools (as in *-copy-biotools*) and the parameter `--db` can be used to copy a PubFetcher database (as in *-copy-db*). If these parameters are omitted, then the entire current bio.tools content is fetched as part of the command (as in *-get-biotools*) and an empty PubFetcher database is initialised (as in *-init-db*).

All parameters (like `--timeout`, `--fetcher-threads`, `--matches`) influencing the step commands can also be specified and are passed on to the steps accepting them.

An example of running the command:

```
$ java -jar path/to/pub2tools-<version>.jar -all results --edam http://edamontology.
↪org/EDAM.owl --idf https://github.com/edamontology/edammap/raw/master/doc/biotools.
↪idf --idf-stemmed https://github.com/edamontology/edammap/raw/master/doc/biotools.
↪stemmed.idf --month 2019-08
```

2.2.7 -resume

This command will run all steps starting with the step stored in *step.txt* until the last step (*-map*). Setup must have been completed separately beforehand. If `step.txt` is missing, then the command will run all steps, starting with *-fetch-pub*.

In general, after a step is successfully completed, the next step is written to `step.txt`. Thus, given some *output directory* where running Pub2Tools has been aborted (but setup has been completed), the `-resume` command allows finishing a Pub2Tools run, while not re-executing already done steps, with just one command.

As no *setup commands* are run, then there are no mandatory parameters. However, if resuming from an interrupted command, then the same parameters that were used for that command should be respecified here.

An example of running the command:

```
$ java -jar path/to/pub2tools-<version>.jar -resume results
```

2.3 Parameters

Parameters give required arguments to or influence the *setup commands* and *steps* and begin with two dashes (`--`). All the `-copy` setup commands have a mandatory parameter specifying the source of the file to be copied. The *-select-pub* setup command needs parameters to specify the data range for fetching publication IDs. All other parameters are optional and influence the default behaviour of the commands.

Parameter	Parameter args	Default	Description
`--edam`	*<file or URL>*		The EDAM ontology OWL file to be copied to the output directory with *-copy-edam* (or *-all*)
`--idf`	*<file or URL>*		The unstemmed IDF file to be copied to the output directory with *-copy-idf* (or *-all*)
`--idf-stemmed`	*<file or URL>*		The stemmed IDF file to be copied to the output directory with *-copy-idf* (or *-all*)
`--biotools`	*<file or URL>*		The JSON file containing the entire bio.tools content to be copied to the output directory with *-copy-biotools* (or *-all*)
`--from`	*<ISO-8601 date>*		The start date (in the form `2019-08-23`) of the date range used to fetch publication IDs from with *-select-pub* (or *-all*)
`--to`	*<ISO-8601 date>*		The end date (in the form `2019-08-23`) of the date range used to fetch publication IDs from with *-select-pub* (or *-all*)
`--month`	*<ISO-8601 month>*		One month (in the form `2019-08`) for which publication IDs should be fetched from with *-select-pub* (or *-all*)
`--day`	*<ISO-8601 date>*		One day (in the form `2019-08-23`) for which publication IDs should be fetched from with *-select-pub* (or *-all*)
`--disable-tool-restriction`			If specified, using phrase combinations to narrow down publication IDs to only those potentially about tools is not done with *-select-pub* (or *-all*)
`--custom-restriction`	*<string>*		Additional restrictions for publication IDs to be fetched with *-select-pub* (or *-all*), specified using the Europe PMC search syntax (https://europepmc.org/searchsyntax)
`--disable-exclusion`			If specified, some further restrictions to eliminate a few wrong publication IDs are not used with *-select-pub* (or *-all*)
`--pub`	*<file or URL>*		The file containing publication IDs to be copied to the output directory with *-copy-pub* (or *-all*)
`--db`	*<file or URL>*		The PubFetcher database file to be copied to the output directory with *-copy-db* (or *-all*)
`--fetcher-threads`	*<integer>*	3	Number of threads to use for parallel fetching in *-fetch-pub* and *-fetch-web* (or *-all* or *-resume*)
`--mapper-threads`	*<integer>*	4	Number of threads to use for parallel mapping in *-map* (or *-all* or *-resume*)
`--verbosity`	*<LogLevel>*		The level of log messages that code called from PubFetcher (like fetching publications and web pages) and EDAMmap (like progress of mapping) can output to the console. For example, a value of `WARN` would enable printing of `ERROR` and `WARN` level log messages from PubFetcher and EDAMmap code. Possible values are `OFF`, `ERROR`, `WARN`, `INFO`, `DEBUG`. To note, this affects only log messages output to the console, as log messages of any level from PubFetcher and EDAMmap code are written to the *log file* in any case.

In addition, some commands are influenced by parameters defined in PubFetcher or EDAMmap: Preprocessing parameters (influences *-pass1*, *-pass2* and *-map*), Fetching parameters (influences *-fetch-pub*, *-fetch-web*, *-pass2* and *-map*) and Mapping parameters (influences *-map*).

Note: The `--stemming` parameter in the Preprocessing parameters is always `false` for *-pass1* and *-pass2*, but it can be set to either `true` or `false` for *-map* (default is `true`).

2.4 Examples

A quickstart example for August 2019, where the EDAM ontology and IDF files and the entire content of bio.tools are downloaded from the web and there are no deviations from default values in any of the steps, is the following:

```
$ java -jar path/to/pub2tools-<version>.jar -all results \
--edam http://edamontology.org/EDAM.owl \
--idf https://github.com/edamontology/edammap/raw/master/doc/biotools.idf \
--idf-stemmed https://github.com/edamontology/edammap/raw/master/doc/biotools.stemmed.
↪idf \
--month 2019-08
```

An example getting potential tools about COVID-19 from the year 2020:

```
$ java -jar path/to/pub2tools-<version>.jar -all results \
--edam http://edamontology.org/EDAM.owl \
--idf https://github.com/edamontology/edammap/raw/master/doc/biotools.idf \
--idf-stemmed https://github.com/edamontology/edammap/raw/master/doc/biotools.stemmed.
↪idf \
--from 2020-01-01 --to 2020-12-31 \
--custom-restriction '"2019-nCoV" OR "2019nCoV" OR "COVID-19" OR "SARS-CoV-2" OR
↪"COVID19" OR "COVID" OR "SARS-nCoV" OR ("wuhan" AND "coronavirus") OR "Coronavirus"␣
↪OR "Corona virus" OR "corona-virus" OR "corona viruses" OR "coronaviruses" OR "SARS-
↪CoV" OR "Orthocoronavirinae" OR "MERS-CoV" OR "Severe Acute Respiratory Syndrome"␣
↪OR "Middle East Respiratory Syndrome" OR ("SARS" AND "virus") OR "soluble ACE2" OR (
↪"ACE2" AND "virus") OR ("ARDS" AND "virus") or ("angiotensin-converting enzyme 2"␣
↪AND "virus")'
```

The next example executes each individual setup and step command from start to final results, while changing some default values:

```
# The EDAM ontology was previously downloaded to the local file system
# to path/to/EDAM.owl and is copied from there to results/EDAM.owl
$ java -jar path/to/pub2tools-<version>.jar -copy-edam results \
--edam path/to/EDAM.owl
# The IDF files have been downloaded to the local file system and are
# copied from there to the output directory "results"
$ java -jar path/to/pub2tools-<version>.jar -copy-idf results \
--idf path/to/tf.idf --idf-stemmed path/to/tf.stemmed.idf
# All bio.tools content is fetched to the file results/biotools.json
$ java -jar path/to/pub2tools-<version>.jar -get-biotools results
# Candidate publication IDs from August 2019 are fetched to the file
# results/pub.txt
$ java -jar path/to/pub2tools-<version>.jar -select-pub results \
--from 2019-08-01 --to 2019-08-31
# An empty PubFetcher database is initialised to results/db.db
```

(continues on next page)

(continued from previous page)

```
$ java -jar path/to/pub2tools-<version>.jar -init-db results
# In the first step, the content of publications listed in
# results/pub.txt is fetched to results/db.db, while the connect and
# read timeout is changed to 30 seconds, some fixes for outdated
# journal scraping rules are loaded from a YAML file and the number
# of threads used for parallel fetching is doubled from the default 8
$ java -jar path/to/pub2tools-<version>.jar -fetch-pub results \
--timeout 30000 --journalsYaml journalsFixes.yaml --fetcher-threads 16
# The first pass of Pub2Tools is run
$ java -jar path/to/pub2tools-<version>.jar -pass1 results
# Web pages extracted by the first pass are fetched, with some default
# parameters modified analogously to -fetch-pub
$ java -jar path/to/pub2tools-<version>.jar -fetch-web results \
--timeout 30000 --webpagesYaml webpagesFixes.yaml --fetcher-threads 16
# The second pass of Pub2Tools is run, culminating in the files
# results/results.csv, results/diff.csv and results/new.json
$ java -jar path/to/pub2tools-<version>.jar -pass2 results
# EDAM annotations are added to results/new.json and output to
# results/to_biotools.json, with stemming turned off, mapping done in
# parallel in 8 threads and up to 5 terms output for all EDAM branches
$ java -jar path/to/pub2tools-<version>.jar -map results \
--stemming false --branches topic operation data format \
--mapper-threads 8
```

The following example is equivalent with the previous one, just all commands have been replaced with one *-all* command:

```
$ java -jar path/to/pub2tools-<version>.jar -all results \
--edam path/to/EDAM.owl --idf path/to/tf.idf \
--idf-stemmed path/to/tf.stemmed.idf --month 2019-08 \
--timeout 30000 --journalsYaml journalsFixes.yaml \
--webpagesYaml webpagesFixes.yaml --fetcher-threads 16 \
--stemming false --branches topic operation data format \
--mapper-threads 8
```

All files of the setup can be obtained through some external means and simply copied to the output directory `results`:

```
# Copy a previously downloaded EDAM ontology to the output directory
$ java -jar path/to/pub2tools-<version>.jar -copy-edam results \
--edam path/to/EDAM.owl
# Copy previously downloaded IDF files to the output directory
$ java -jar path/to/pub2tools-<version>.jar -copy-idf results \
--idf path/to/tf.idf --idf-stemmed path/to/tf.stemmed.idf
# Copy the existing content of bio.tools in JSON format, obtained
# through a different tool or through a previous run of Pub2Tools
# to the output directory
$ java -jar path/to/pub2tools-<version>.jar -copy-biotools results \
--biotools path/to/biotools.json
# Copy publication IDs obtained through some different means, for
# example a small list of manually entered IDs meant for testing,
# to the output directory
$ java -jar path/to/pub2tools-<version>.jar -copy-pub results \
--pub path/to/pub.txt
# Copy a PubFetcher database preloaded with potentially useful
# content to the output directory
$ java -jar path/to/pub2tools-<version>.jar -copy-db results \
```

(continues on next page)

(continued from previous page)

```
--db path/to/db.db
# We can use the -resume command to run all step commands in one go;
# the number of threads has been doubled from default values and up to
# 5 EDAM terms are output in the mapping step for the default branches
# of "topic" and "operation"
$ java -jar path/to/pub2tools-<version>.jar -resume results \
--fetcher-threads 16 --mapper-threads 8
```

The following *-all* command is equivalent to the previous list of commands:

```
$ java -jar path/to/pub2tools-<version>.jar -all results \
--edam path/to/EDAM.owl --idf path/to/tf.idf --idf-stemmed \
path/to/tf.stemmed.idf --biotools path/to/biotools.json \
--pub path/to/pub.txt --db path/to/db.db
--fetcher-threads 16 --mapper-threads 8 ^C (Interrupted)
# But for some reason, the -all command was interrupted. If this
# happened during a step command (when all setup was already done),
# then finishing the run can be done with the -resume command. The
# process is resumed by restarting the step that was interrupted and
# running the remaining steps up to the end. The same step parameters
# that were supplied to -all must also be supplied to -resume.
$ java -jar path/to/pub2tools-<version>.jar -resume results \
--fetcher-threads 16 --mapper-threads 8
```

When fetching publications and web pages, some resources might be temporarily down. So for slightly better results, one option could be to wait a few days after an initial fetch and hope that a few extra resources would be available then. Due to PubFetcher's logic, publications and web pages that were successfully fetched in full the first time, are not retried during refetching:

```
$ java -jar path/to/pub2tools-<version>.jar -fetch-pub results
$ java -jar path/to/pub2tools-<version>.jar -pass1 results
$ java -jar path/to/pub2tools-<version>.jar -fetch-web results
$ # Wait a few days
$ java -jar path/to/pub2tools-<version>.jar -fetch-pub results
$ java -jar path/to/pub2tools-<version>.jar -pass1 results
$ java -jar path/to/pub2tools-<version>.jar -fetch-web results
$ java -jar path/to/pub2tools-<version>.jar -pass2 results
$ java -jar path/to/pub2tools-<version>.jar -map results
```

To run all steps again after the wait, another option would be to just use the *-resume* command after removing *step.txt*:

```
$ java -jar path/to/pub2tools-<version>.jar -fetch-pub results
$ java -jar path/to/pub2tools-<version>.jar -pass1 results
$ java -jar path/to/pub2tools-<version>.jar -fetch-web results
$ # Wait a few days
$ rm results/step.txt
$ java -jar path/to/pub2tools-<version>.jar -resume results ^C (Interrupted)
# The -resume command was interrupted for some reason. If -resume is
# now run again, it will not start again from -fetch-pub, but from the
# step that was interrupted.
$ java -jar path/to/pub2tools-<version>.jar -resume results
```

Note: Before a bigger run of Pub2Tools, it could be beneficial to test if scraping rules are still up to date. The running of Pub2Tools in a network with good access to journal articles could also be beneficial, as publisher web sites have to be consulted sometimes.

2.4.1 Improving existing bio.tools entries

When Pub2Tools is run on a relatively new batch of publication IDs, then most results will end up in *to_biotools.json* as new entry suggestions for bio.tools and only a few results will be diverted to *diff.csv* as fix suggestions of existing bio.tools entries. This is expected, as most new articles will be about tools not seen before and only some will be update articles of tools already entered to bio.tools or articles of tools entered to bio.tools through some other means than Pub2Tools.

But as an alternative and additional example of Pub2Tools usage we can consider the following: let's get all publication IDs (and nothing else) currently in bio.tools and run Pub2Tools on these IDs. Now, most results will end up in *diff.csv*, as all entries are determined to be already *existing in bio.tools*. So what we get out of this, is a large spreadsheet of suggestions (`diff.csv`) on what to improve in existing bio.tools content. A lot of the suggestions are incorrect, but there should also be many valuable fixes and additions there. If content missing in bio.tools was found, then the tool will suggest adding it to bio.tools (to "publication", "link", "download", "documentation", "language" or "credit"), or for some content types, the tool will suggest modifying existing content (in "name", "homepage", "license" or "credit"). No suggestions (for removal) are given for content that is present in bio.tools, but that was not found by the tool. All values suggested by the tool are valid according to biotoolsSchema, but they don't necessarily follow the curation guidelines.

In addition, the file *to_biotools.json* will probably not be totally empty, but contain a few suggested new entries to bio.tools – this happens, because some publications seem to be about multiple tools and are broken up, and it would seem that some of these broken up tools are missing in bio.tools. As detecting multiple tools from one publication is relatively crude right now, then the quality of these new entries in the JSON file might not be the best, but going through them might still be worth the while.

Running Pub2Tools on existing bio.tools content can be done in the following way (needs EDAMmap-Util from EDAMmap):

```
# Download all bio.tools content to biotools.json
$ java -jar path/to/edammap-util-<version>.jar -biotools-full biotools.json
# Extract all publication IDs present in biotools.json to pub.txt
$ java -jar path/to/edammap-util-<version>.jar -pub-query biotools.json \
--query-type biotools -txt-ids-pub pub.txt --plain
# Run Pub2Tools with default parameters, outputting all to directory "results"
$ java -jar path/to/pub2tools-<version>.jar -all results \
--edam http://edamontology.org/EDAM.owl \
--idf https://github.com/edamontology/edammap/raw/master/doc/biotools.idf \
--idf-stemmed https://github.com/edamontology/edammap/raw/master/doc/biotools.stemmed.
↪idf \
--biotools biotools.json \
--pub pub.txt
```

Note: The example works best when still only few entries have been added to bio.tools from curated Pub2Tools results, as for entries from these results mistakes made by Pub2Tools would have to be gone through repeatedly.

CHAPTER 3

Output

All results and intermediate files will be output to the *output directory* specified by the user. The main results files of interest are *results.csv*, where all results with all possible extra data are output, *diff.csv*, where potential fixes to some bio.tools content is output (based on differences of new entries and corresponding entries already *existing in bio.tools*), and *to_biotools.json*, where new additions to bio.tools are output. All intermediate files are also copied and kept in the output directory for potential reproducibility and debugging purposes.

The output of Pub2Tools does not lead to fully automatic growth and improvement of bio.tools content. The results have to be curated: filled attributes might need checking or editing (especially *description*), some suggested entries might actually not be tools or otherwise not be suitable for bio.tools, information for many attributes is not found (as seen in *performance*), fixes in `diff.csv` have to be checked and applied manually, etc.

3.1 Output directory

The name of the output directory is chosen by the user and must be specified after all commands, like `-copy-edam directory_name` or `-pass1 directory_name`. In case the output directory is not existing, it will be created. And while the name of the output directory can be chosen, the names of the results and intermediate files are fixed. Short descriptions of these files is given below.

3.1.1 pub2tools.log

The log file is created to a new output directory by the first command that is run with the output directory as argument. The first two lines of the log file are the arguments given to Pub2Tools and the version of Pub2Tools. Then, the same lines that are output to the console while the command is running, are output to the log file, except also all `DEBUG` level messages are output and all log messages coming from PubFetcher and EDAMmap code are output to the log file (outputting those to the console is turned off by default with the *parameter* `--verbose OFF`). Any subsequent commands run on the output directory will just append to the existing log file.

The log file could later be used for debugging. One option would be analysing `ERROR` level messages, for example with `grep ERROR pub2tools.log | less`. More information about the structure of a log line and analysing the logs can be found in the PubFetcher documentation about the log file.

3.1.2 EDAM.owl

The EDAM ontology file is needed by the *-map* step to add EDAM terms to the results. It can be downloaded from the EDAM ontology GitHub and copied to the output directory with the setup command *-copy-edam*.

3.1.3 tf.idf

A simple file of tab-separated values containing normalised tf–idf scores of (unstemmed) words occuring in the bio.tools corpus. It can either be downloaded or generated, more information at *-copy-idf*, which is the command used to copy the file to the output directory.

Note: As the IDF files are one of the largest files in the output directory and these files are potentially equal across many runs of Pub2Tools, then these could be the files to delete first in a finalised output directory to save disk space (or linked to some master IDF files with `ln -s`).

3.1.4 tf.stemmed.idf

Like *tf.idf*, except the words are stemmed.

3.1.5 biotools.json

The entire content of bio.tools in JSON format and adhering to biotoolsSchema, either downloaded with *-get-biotools* or copied with *-copy-biotools*.

3.1.6 pub.txt

A list of candidate publication IDs to search tools from. It's a simple text file containing publication IDs in the form `<pmid>\t<pmcid>\t<doi>`, one per line. Empty lines and lines beginning with `#` are ignored. It can be either fetched with *-select-pub* or copied with *-copy-pub* or created manually.

3.1.7 db.db

A PubFetcher database file containing the contents of publications and web pages fetched as part of a Pub2Tools run. It needs to be initialised with *-init-db* or a database with prefetched content can be copied with *-copy-db*. The database can be queried or manipulated with PubFetcher-CLI or EDAMmap-Util.

3.1.8 step.txt

Used to keep track of the current *step* being run. Can contain the value `None`, `-fetch-pub`, `-pass1`, `-fetch-web`, `-pass2`, `-map` or `Done`. The value indicates, which step should be run next. So for example, after *-pass1* completes successfully, the value `-fetch-web` will be written to the file. If the file is present and contains any value other than `None`, then this means that some steps have been run and no *Setup commands* can be run anymore. But the main use of the file is enabling the *-resume* command: when that command is run it checks which step should be run next and runs that step and all subsequent steps until the last step of *-map* is completed and `Done` is written out.

3.1.9 web.txt

A list of webpage URLs extracted from the publication abstracts and fulltexts by the *-pass1* command that are matching the (up to 5 per publication) names suggested for the tools the publications are potentially about. These URLs are candidates for the tool homepage and other link attributes in bio.tools and the content of these links needs to be fetched in *-fetch-web*. The URLs are simply written one per line, with empty lines and lines beginning with # being ignored.

3.1.10 doc.txt

Same as *web.txt*, except links determined to be about documentation are written here instead (because the PubFetcher database has a separate store for docs).

3.1.11 pass1.json

Results of the *-pass1* command, that are later used as input for *-pass2*. The results include information about the publication (like its IDs, title, publication date and journal, number of citation and corresponding authors) and about the up to 5 candidate names for the potential tool the publication is about (including the name in processed form, the score assigned to the name and links attached to it). Most of the values passed on to *-pass2* also end up in *results.csv*, so more thorough documentation about these values can be found in *results.csv columns*.

3.1.12 results.csv

This file will contain all results of Pub2Tools as output by the *-pass2* command, including entries that were excluded for entry to bio.tools or found to be already existing there. In addition to the end results that can be inserted to bio.tools attributes, each entry will contain all possible other data related to the entry and values of intermediate results, but also values currently present in bio.tools for entries that were found to be existing there. All these values are documented in *results.csv columns*. The first row of the file specifies the column names and the second row contains links to the column documentations in *results.csv columns*.

3.1.13 diff.csv

This file will contain entries that were found to be *existing in bio.tools* in *-pass2*. More precisely, it will only contain entries, that were found to be existing in bio.tools and for which some value was found to be different or missing in bio.tools, and the contents of the file will be a listing of these difference (i.e. differing or missing values). Many of these differences are mistakes made by Pub2Tools, but many are also pointing to incorrect or missing information in bio.tools, thus the contents of this file can be used to improve existing entries of bio.tools. In rare circumstances, some entries that are not actually already existing in bio.tools might be mistakenly diverted here (instead of *to_biotools.json*) – such entries should be added to bio.tools manually. This file can be especially useful if Pub2Tools is run on all publications currently in bio.tools, like exemplified in *Improving existing bio.tools entries*. The structure of the file is documented in *diff.csv columns*. The first row of the file specifies the column names and the second row contains links to the column documentations in *diff.csv columns*.

3.1.14 new.json

This file will contain all new entries suggested for addition to bio.tools, as *decided* and output by *-pass2* and adhering to biotoolsSchema. The file is fed as input to the command *-map*, producing *to_biotools.json*, which is the file that should actually be used to add the new entries to bio.tools.

The following bio.tools attributes will always be filled: name attribute, description attribute (if nothing else is found, then it is filled with the publication abstract), homepage attribute (if no links found, then filled with a link to the publication itself) and publication attribute. Additionally, an effort is made to fill the following attributes: language attribute, license attribute, link attribute, download attribute, documentation attribute and credit attribute. Further information about possible values of these attributes (for example about the messages to the curator in the *description*) can be found in *to_biotools.json attributes*.

3.1.15 map.txt

Additional data about the EDAMmap results got using the *-map* command, in plain text format.

3.1.16 map/

Additional data about the EDAMmap results got using the *-map* command, in a directory of HTML files. To see this mapping data, open `map/index.html` in a web browser.

3.1.17 map.json

Additional data about the EDAMmap results got using the *-map* command, in JSON format.

3.1.18 to_biotools.json

Same as *new.json*, except EDAMmap terms have been added by the *-map* command to the function attribute and topic attribute. This is the file that should be used to add new entries to bio.tools. Rarely, some entries here are actually already existing in bio.tools (and thus should have been output to *diff.csv* instead) – such entries should evidently not be added to bio.tools (however, such entries might still contain useful information on what to change in those existing entries). Further information about possible values of the attributes can be found in *to_biotools.json attributes*.

3.2 results.csv columns

pmid As results are extracted from publications, then the first 3 columns are the IDs of the publication – here, the PubMed ID of the publication is output. These publication IDs are used to fill the publication attribute of bio.tools. Sometimes, multiple publications seem to be about the same tool – in that case the corresponding results are *merged into one row* and the PubMed IDs of these different publications will be separated by `" | "` here.

pmcid Like *pmid*, but for the PubMed Central ID of publications.

doi Like *pmid*, but for the Digital Object Identifier (DOI) of publications.

same_suggestions Currently, results got from two different publications are *merged into one result*, if their top name *suggestion* is exactly equal and *confidence* is not "very low". If the names are equal, but confidence of at least one of the names is "very low", then the publications are not merged, but instead linked through this column (where one result will contain publication IDs of the other result and vice versa). If multiple such links are made, then the publication IDs of the different linked results are separated by `" | "`.

score The goodness score of the *suggestion* is calculated in the first pass (*-pass1*) and shows *confidence* in the extracted tool name (and not in how "good" or high impact the tool itself is). Entries in the results file are sorted by score (for entries whose score is at least 1000), but there are a few other things to consider in assessing whether an entry is about a tool and suitable for suggestion to bio.tools – whether an entry is suggested can be seen in the *include* column.

score2 If *score* is lower than 1000, then this *second score is calculated* in the second pass (*-pass2*) for further fine-tuning of entries of lower confidence. Entries that have this second score are sorted by it instead of *score*.

score2_parts Values of the *four parts* of *score2*. Summing these four parts, plus the value of *score*, will get as result *score2*.

confidence A confidence of "high", "medium", "low" or "very low" *is determined* based on the values of *score* and *score2*.

include `true`, if the *final decision* of Pub2Tools, based on some additional aspects in addition to *score* and *score2*, is that the entry is about a tool. In the `true` case, the entry will be suggested as a new tool to add to bio.tools, unless the value in the *existing* column is not empty. Also, if *confidence* is "very low", but *include* is still `true`, then the entry is quite possibly about a tool and suggested for entry, however, the confidence in the tool name *suggestion* is very low and should be checked.

existing Will contain bio.tools ID(s) of entries that are found to be already *existing in bio.tools*. If multiple entries in bio.tools are matched, then the IDs are separated by `" | "`. Entries that are found to be already existing in bio.tools are not suggested as new tools, however, if there are differences in information currently in bio.tools and information extracted by Pub2Tools for these entries, then these differences are highlighted in *diff.csv* (and for entries that were found to be existing due to matching publication IDs in bio.tools, entry to *diff.csv* is done even if *include* is `false`).

suggestion_original The name suggested for the tool, in original form as extracted from the title and abstract of the publication. As there are syntactic restrictions and a limited set of characters allowed in the name (latin letters, numbers and some punctuation symbols, as seen in name attribute API docs), then for some entries the original suggestion must be edited: invalid characters are either replaced (done for accents, greek letters, etc) or discarded altogether and too long suggestions truncated. Only syntactic rules mandated by biotoolsSchema are followed, curation guidelines for the name attribute are not necessarily followed. The value in this column will be empty, if no such modifications need to be made, otherwise this column will contain the original name and the *suggestion* column the modified form of the name.

suggestion The name suggested as the name attribute of the tool for bio.tools, extracted from the title and abstract of the publication in the first pass (*-pass1*).

suggestion_processed A further processed version of *suggestion* (with letters converted to lowercase and symbols removed), used in many parts of the Pub2Tools algorithm (like matching the name to extracted links).

publication_and_name_existing Contains bio.tools IDs (separated by `" | "`) of entries in bio.tools that have exactly the same name and whose publications are also present in this entry constructed by Pub2Tools. Matching publication IDs mean that the entry is considered *existing in bio.tools* and it is added to the *existing* column (even if *include* is `false`).

name_existing_some_publication_different Contains bio.tools IDs (separated by `" | "`) of entries in bio.tools that have exactly the same name and for which some publications are also present in this entry constructed by Pub2Tools, but some are not (IDs of publications found by Pub2Tools but not present in bio.tools are written in parenthesis after the bio.tools ID, with possible multiple publications separated by `" ; "`). Some matching publication IDs mean that the entry is considered *existing in bio.tools* and it is added to the *existing* column (even if *include* is `false`).

some_publication_existing_name_different Contains bio.tools IDs (separated by `" | "`) of entries in bio.tools whose publications are also present in this entry constructed by Pub2Tools, but whose name is different than the name found by Pub2Tools (the tool name of the entry in bio.tools is written in parenthesis after the ID; in addition, if Pub2Tools has found publications that are not present in the matching bio.tools entry, then the IDs of these publications are written to another set of parenthesis after the ID and name, with potential multiple publications separated by `" ; "`). Some matching publication IDs mean that the entry is considered *existing in bio.tools* and it is added to the *existing* column (even if *include* is `false`). The difference in name is highlighted in *diff.csv*.

name_existing_publication_different Contains bio.tools IDs (separated by `" | "`) of entries in bio.tools that have

exactly the same name as this entry constructed by Pub2Tools, but that have no matching publication IDs with this entry (publications found by Pub2Tools are written in parenthesis after the bio.tools ID, with possible multiple publications separated by `" ; "`). The new entry is considered *existing in bio.tools* only if one of the bio.tools IDs in this column also occurs in the *link_match* column or if a *credit* of the new entry matches a credit in a bio.tools entry corresponding to these bio.tools IDs (and additionally, *confidence* must not be "very low" and *include* must be `true`), in which case bio.tools IDs matching these criterias are added to the *existing* column.

name_match Like *name_existing_publication_different*, except the name of the bio.tools entry is not exactly equal to the name of the new entry constructed by Pub2Tools, just their processed names are equal (the processed name being like in *suggestion_processed* but with potential version information removed from the end). Also, non-matching publication IDs will not be output in parenthesis after the bio.tools ID – the name of the tool in bio.tools will be output instead.

link_match Contains bio.tools IDs (separated by `" | "`) of entries in bio.tools that have any matching link with any link extracted by Pub2Tools for this *suggestion* (as seen in *links_abstract* or *links_fulltext*). Links don't have to be equal: in addition to the standard `www` and `index.html` parts, the lowest subdomain and last path of the links are ignored when matching. The common matching part of the matching link is output in parenthesis after the bio.tools ID, with potential multiple partial links separated by `" ; "`. This column is not filled with bio.tools IDs already occuring in *publication_and_name_existing*, *name_existing_some_publication_different* or *some_publication_existing_name_different*. If any of the bio.tools IDs occuring here also occur in *name_existing_publication_different* or *name_match*, then this entry is considered *existing in bio.tools* and these common bio.tools IDs are added to the *existing* column.

name_word_match Contains bio.tools IDs (separated by `" | "`) of entries in bio.tools whose name has a matching word with a word from the name of this entry constructed by Pub2Tools. The name of the entry in bio.tools follows in parenthesis. If a bio.tools ID is already in any of the columns from *publication_and_name_existing* to *link_match*, then it is not added here. Also, if too many bio.tools IDs would be added (over 5), then nothing is output here. The values in this column are not used anywhere in the Pub2Tools algorithm.

links_abstract Contains URLs (separated by `" | "`) extracted from the abstracts of publications and matched to the *suggestion*. This is done in the first pass (*-pass1*).

links_fulltext Contains URLs (separated by `" | "`) extracted from the full texts of publications and matched to the *suggestion*. This is done in the first pass (*-pass1*).

from_abstract_link `true`, if the tool name in *suggestion* was extracted from a link in the publication abstract (as that name was only occuring in a link and not elsewhere in the text of the abstract or title). If there are *other_suggestions*, then the Boolean values (separated by `" | "`) for those will be appended after `" | "`.

homepage A URL suggested as the homepage attribute of the tool for bio.tools. The homepage is selected when *dividing links* (i.e. the links in *links_abstract* and *links_fulltext* are divided) in the second pass (*-pass2*).

homepage_broken `true`, if the homepage link seems to be broken. A broken page is suggested as the homepage, as no better alternatives were found. The broken status of a web page is determined in PubFetcher code called by Pub2Tools based on reachability and the HTTP status code.

Note: A reportedly broken homepage can sometimes still be functional (for example, maybe it was temporarily down at the time Pub2Tools was run) – this could be manually checked in a web browser.

---

homepage_missing `true`, if no links (even broken ones) matching the *suggestion* were found, i.e. a homepage could not be extracted. In that case, the *homepage* column is still filled, but with a link to the publication. A missing homepage does not necessarily mean that the entry is not a tool, it just means that no suitable links in the publication abstract or fulltext were matched to the extracted tool name in *suggestion* (either Pub2Tools failed to find the homepage or the publication just doesn't mention any links of the tool).

homepage_biotools Contains homepages (separated by `" | "`) of the bio.tools entries corresponding to the bio.tools

IDs in *existing*, that is, if the current entry constructed by Pub2Tools is found to be existing in bio.tools, then the homepage currently in bio.tools is output here to contrast with the value in the column *homepage*. If a homepage currently in bio.tools is determined to be broken by Pub2Tools, then `" (broken)"` will follow the homepage URL and in addition, if the homepage is determined to be problematic in bio.tools itself, then `" (homepage_status: x)"` will follow the homepage URL (where `x` is a status number other than `0`, as got through the bio.tools API).

link A list of URLs (separated by `" | "`) suggested for the link attribute of the tool for bio.tools. These links are selected when *dividing links* (the links in *links_abstract* and *links_fulltext*) in the second pass (*-pass2*). After each URL, the type of the link will follow in parenthesis (in case of the link attribute, for example "Repository" or "Mailing list").

link_biotools Contains lists (separated by `" | "`) of links (separated by `" ; "`) of the bio.tools entries corresponding to the bio.tools IDs in *existing*, that is, if the current entry constructed by Pub2Tools is found to be existing in bio.tools, then the links currently in bio.tools are output here to contrast with the values in the column *link*. After each URL, the type of the link will follow in parenthesis (in case of the link attribute, for example "Repository" or "Mailing list").

download Like *link*, except links meant for the download attribute of bio.tools are output.

download_biotools Like *link_biotools*, except download attribute links of existing bio.tools entries are output.

documentation Like *link*, except links meant for the documentation attribute of bio.tools are output.

documentation_biotools Like *link_biotools*, except documentation attribute links of existing bio.tools entries are output.

broken_links Contains link attribute, download attribute and documentation attribute URLs (separated by `" | "`) that were found to be broken when *dividing links* (the links in *links_abstract* and *links_fulltext*) in the second pass (*-pass2*). After each URL, the type of the link will follow in parenthesis (in case of the link attribute, for example "Repository" or "Mailing list"). Links occuring here will not be output to *link*, *download* and *documentation* (and thus not suggested for input to bio.tools), however, if the *homepage is broken*, then the homepage URL will appear both here and in the *homepage* column.

other_scores The rounded scores (separated by `" | "`) of *other_suggestions*, analogous to the *score* column of the main suggestion.

other_scores2 The rounded second scores (separated by `" | "`) of *other_suggestions*, analogous to the *score2* column of the main suggestion.

other_scores2_parts The parts of the rounded second scores (separated by `" | "`) of *other_suggestions*, analogous to the *score2_parts* column of the main suggestion.

other_suggestions_original The unedited names (separated by `" | "`) of *other_suggestions*, analogous to the *suggestion_original* column of the main suggestion.

other_suggestions Up to 4 alternative suggestions for the tool name are extracted in the first pass (*-pass1*). The order of these suggestions was possibly changed (with one of them possibly even elevated to be the main *suggestion*) when *score2 was calculated* in the second pass (*-pass2*). There may also be no alternative suggestions, which shows higher confidence in the main *suggestion*. This column contains the names (for the name attribute of bio.tools) of these alternative suggestions (separated by `" | "`). Alternative suggestions are not suggested for entry to bio.tools, however a message in the *description* will draw the attention of the curator to the existence of possible alternative names of the tool.

other_suggestions_processed The processed names (separated by `" | "`) of *other_suggestions*, analogous to the *suggestion_processed* column of the main suggestion.

other_publication_and_name_existing A column analogous to *publication_and_name_existing*, but for *other_suggestions*. Values of different suggestions are separated by `" | "` and IDs within a suggestion are separated by `" ; "`.

other_name_existing_some_publication_different A column analogous to *name_existing_some_publication_different*, but for *other_suggestions*. Values of different suggestions are separated by `" | "` and IDs within a suggestion are separated by `" ; "`.

other_some_publication_existing_name_different A column analogous to *some_publication_existing_name_different*, but for *other_suggestions*. Values of different suggestions are separated by `" | "` and IDs within a suggestion are separated by `" ; "`.

other_name_existing_publication_different A column analogous to *name_existing_publication_different*, but for *other_suggestions*. Values of different suggestions are separated by `" | "` and IDs within a suggestion are separated by `" ; "`.

other_links_abstract Contains links found in the publication abstract that are matching *other_suggestions*. Links of different suggestions are separated by `" | "` and links within a suggestion are separated by `" ; "`.

other_links_fulltext Contains links found in the publication fulltext that are matching *other_suggestions*. Links of different suggestions are separated by `" | "` and links within a suggestion are separated by `" ; "`.

leftover_links_abstract Contains all links (separated by `" | "`) that were extracted from the publication abstract, but not matched to the main *suggestion* (thus, not output to the *links_abstract* column) or to any *other_suggestions* (thus, not output to the *other_links_abstract* column). These links are just output to this column and not used anywhere else in Pub2Tools.

leftover_links_fulltext Contains all links (separated by `" | "`) that were extracted from the publication fulltext, but not matched to the main *suggestion* (thus, not output to the *links_fulltext* column) or to any *other_suggestions* (thus, not output to the *other_links_fulltext* column). These links are just output to this column and not used anywhere else in Pub2Tools.

title Contains the title(s) of the publication(s) (separated by `" | "`).

tool_title_others Contains the other *tool_title* of a publication that was split into two entries (base on a `" and "`, `" & "` or `", "` in the entire tool_title part of a publication title). If a publication is split into more than two entries, then the other tool_titles will be separated by `" ; "`. If the entry has more than one publication, than the other tool_titles of different publications are separated by `" | "`. Keeping track of these other tool_titles is needed, because if a publication is split into many entries, then all these entries will have a common publication and Pub2Tools would otherwise suggest merging them back into one entry in *diff.csv*.

tool_title_extracted_original The *tool_title* as originally extracted from the publication title. If no *tool_title* can be extracted from the publication title, then this column will be empty. Note, that some processing steps have still been done, for example, other tool_titles have been separated to *tool_title_others*, whitespace has been normalised, some punctuation removed from the start and end of words, etc. This form of the *tool_title* is used as part of the *calculations of the score2* part concerning the *tool_title*.

tool_title The tool_title is the part of the publication title that precedes `": "`, `" - "`, `", a"`, etc. The tool_titles of different publications are separated by `" | "`. In this column, the intermediate extraction step of the tool_title, as presented in *tool_title_extracted_original*, is further processed, for example stop words are removed (this can be further influenced by Preprocessing parameters). Also, if *tool_title_extracted_original* contains an acronym in parenthesis, then this acronym is removed (to *tool_title_acronym*). If this processing does not alter the value in *tool_title_extracted_original*, then the value in this column is left empty for readability purposes. The tool_title is often equal to the name of the tool and thus often (but not always) ends up as the name of the entry in *suggestion*.

tool_title_pruned A further processed *tool_title*, where version information and some common words (like "database", "server", "pipeline") have been pruned. If this pruning doesn't remove anything and thus the value is equal to *tool_title*, then an empty string would be output to this column instead. Like *tool_title_extracted_original*, the pruned version of tool_title is used in the *calculations of the score2* part concerning the tool_title.

tool_title_acronym Contains the acronym version of the *tool_title*, with values of different publications separated by `" | "`. The acronym must be in parenthesis after the expanded name and it is found and extracted when

processing *tool_title_extracted_original*. Like *tool_title_extracted_original* and *tool_title_pruned*, the acronym version of tool_title is used in the *calculations of the score2* part concerning the tool_title.

description A list of descriptions (separated by `"\n\n"`) suggested as the description attribute of the tool for bio.tools. This is the one column that definitely need curation: a curator can choose one of the descriptions from the list or combine multiple description suggestions into the final description of the tool in bio.tools. More information can be found in the *description part* of the second pass (*-pass2*), where the descriptions are constructed.

In addition to the list of descriptions, a list of messages to the curator (also separated by `"\n\n"`) are appended to the descriptions (after a `"\n\n"`). The messages start with `"|||"` and are uppercase. If there are any messages to the curator, then these should be acknowledged, potentially acted upon and deleted. Messages could be the following:

- NOT INCLUDED! (*include* is `false`)
- HOMEPAGE BROKEN! (*homepage_broken* is `true`)
- HOMEPAGE MISSING! (*homepage_missing* is `true`)
- EXISTING AS *publication_and_name_existing*
- EXISTING AS (SOME PUB. MISSING) *name_existing_some_publication_different*
- TOOL (*suggestion*) EXISTING UNDER DIFFERENT NAME AS *some_publication_existing_name_different* (limited to 5; names follow bio.tools IDs in parenthesis)
- NAME EQUAL TO (PUB. DIFFERENT) *name_existing_publication_different*
- NAME (*suggestion*) SIMILAR TO (PUB. DIFFERENT) *name_match* (names follow bio.tools IDs in parenthesis)
- COMMON LINK WITH (PUB. & NAME DIFFERENT) *link_match* (only output if no more than 5 matches; common link parts follow bio.tools IDs in parenthesis)
- CORRECT NAME OF TOOL COULD ALSO BE *other_suggestions* (up to 4; IDs of current bio.tools entries with publications matching the publications of alternative suggestions follow the names of alternative suggestions in parenthesis)

description_biotools Contains the values of the description attributes (separated by `" | "`) of the bio.tools entries corresponding to the bio.tools IDs in *existing*, that is, if the current entry constructed by Pub2Tools is found to be existing in bio.tools, then the descriptions currently in bio.tools are output here to contrast with the value in the column *description*. Line breaks and tabs in the bio.tools description will be replaced with the strings `"\n"`, `"\r"`, `"\t"`.

license_homepage Contains the value of the license field of the PubFetcher webpage corresponding to the *homepage* URL. Nothing is output, if the field is empty – the field can usually be filled when it's a URL of a repository. The license string is output as got from PubFetcher and needs to be mapped to a valid bio.tools *license Enum value* in the second pass (*-pass2*).

license_link Contains the non-empty values (separated by `" | "`) of the license fields of the PubFetcher webpages corresponding to the *link* URLs. The URL follows the license string in parenthesis. The license strings are output as got from PubFetcher and need to be mapped to valid bio.tools *license Enum values* in the second pass (*-pass2*).

license_download Like *license_link*, but for licenses from *download* URLs.

license_documentation Like *license_link*, but for licenses from *documentation* URLs.

license_abstract Contains all bio.tools licenses found from the abstracts of the publications of this entry. Licenses found from one publication abstract are separated by `" ; "` and values from different publications are separated by `" | "`. The publication IDs of the abstract where a license was found will follow the license value. The *license value* is extracted in the second pass (*-pass2*).

license The license suggested as the value of the license attribute of the tool for bio.tools. This license value is chosen as the most common value occuring among the values of *license_homepage*, *license_link*, *license_download*, *license_documentation* and *license_abstract*. URLs and publication IDs (separated by `", "`) of the webpages and abstracts where the chosen license was encountered will follow the *license value* in parenthesis.

license_biotools Contains the values of the license attribute (separated by `" | "`) of the bio.tools entries corresponding to the bio.tools IDs in *existing*, that is, if the current entry constructed by Pub2Tools is found to be existing in bio.tools, then the licenses currently in bio.tools are output here to contrast with the value in the column *license*.

language_homepage Contains the value of the language field of the PubFetcher webpage corresponding to the *homepage* URL. Nothing is output, if the field is empty – the field can usually be filled when it's a URL of a repository. The language value is output as got from PubFetcher and needs to be mapped to valid bio.tools *language Enum value(s)* in the second pass (*-pass2*).

language_link Contains the non-empty values (separated by `" | "`) of the language fields of the PubFetcher webpages corresponding to the *link* URLs. The URL follows the language value in parenthesis. The language value is output as got from PubFetcher and needs to be mapped to valid bio.tools *language Enum values* in the second pass (*-pass2*).

language_download Like *language_link*, but for licenses from *download* URLs.

language_documentation Like *language_link*, but for licenses from *documentation* URLs.

language_abstract Contains all bio.tools languages found from the abstracts of the publications of this entry. Languages found from one publication abstract are separated by `" ; "` and values from different publications are separated by `" | "`. The publication IDs of the abstract where a language was found will follow the language value. The *language value* is extracted in the second pass (*-pass2*).

language The languages (separated by `" ; "`) suggested as the content of the language attribute of the tool for bio.tools. The languages are put together from all language values found in *language_homepage*, *language_link*, *language_download*, *language_documentation* and *language_abstract* (duplicate values are merged). URLs and publication IDs (separated by `", "`) of the webpages and abstracts where a language was encountered will follow each *language value* in parenthesis.

language_biotools Contains the values of the language attribute of the bio.tools entries corresponding to the bio.tools IDs in *existing*, that is, if the current entry constructed by Pub2Tools is found to be existing in bio.tools, then the languages currently in bio.tools are output here to contrast with the values in the column *language*. Languages of a bio.tools entry are separated by `" ; "` and languages of different entries are separated by `" | "`.

oa `true`, if the publication is Open Access (according to the PubFetcher's oa field of the publication). Values of different publication are separated by `" | "`. This information is just got as a side effect of fetching publications in *-fetch-pub* and it is not used anywhere in Pub2Tools.

journal_title Journal titles of publications (separated by `" | "`) as got from the PubFetcher journalTitle field. Journal titles are used as part of the publication IDs selection process in *-select-pub* and in excluding a few publications from certain journals.

pub_date Publication dates of publications (separated by `" | "`) as got from the PubFetcher pubDateHuman field (the value of the pubDate field follows in parenthesis). The publication date is the date of first publication, whichever is first, electronic or print publication, which is not the same as the "CREATION_DATE" used in *-select-pub*. Therefore, if Pub2Tools is run for some concrete month (using `--month`), then not all publications will necessarily have a publication date from that month (it can be from a previous month, but for some upcoming publications also from a future month). Currently, the publication date is used only to calculate *citations_count_normalised*.

citations_count Numbers (separated by `" | "`) showing how many times publications have been cited as got from the PubFetcher citationsCount field. This information is obtained from Europe PMC, which usually has lower numbers than other citation databases. Furthermore, if Pub2Tools is run on recent publications, then the value is usually `0`, as not enough time has passed for others to cite the articles. The count can be normalised by *pub_date*, giving the value in *citations_count_normalised*.

citations_timestamp The timestamps (separated by `" | "`) when *citations_count* of publications were last updated as got from the PubFetcher citationsTimestampHuman field (the value of the citationsTimestamp field follows in parenthesis). Used when calculating *citations_count_normalised*.

citations_count_normalised The *citations_count* normalised by *pub_date*. The exact formula is `citations_count / (citations_timestamp - pub_date) * 1000000000`, where the unit of `citations_timestamp` and `pub_date` is milliseconds (since Unix epoch). Currently, the result is not used anywhere in Pub2Tools, but it might be useful for prioritising or selecting candidates from a large batch of older publications.

corresp_author_name Names of the corresponding authors of the publications as got from the PubFetcher correspAuthor field. The names of corresponding authors of a publication are separated by `" ; "` and values from different publications are separated by `" | "`.

credit_name_biotools Contains the values of the credit name attribute of the credit group of the bio.tools entries corresponding to the bio.tools IDs in *existing*, that is, if the current entry constructed by Pub2Tools is found to be existing in bio.tools, then the credit names currently in bio.tools are output here to contrast with the values in the column *corresp_author_name*. Values of different credit name attributes of a bio.tools entry are separated by `" ; "` and values from different bio.tools entries are separated by `" | "`.

corresp_author_orcid Like *corresp_author_name*, but for ORCID iDs of corresponding authors.

credit_orcidid_biotools Like *credit_name_biotools*, but for the ORCID iD attribute.

corresp_author_email Like *corresp_author_name*, but for e-mails of corresponding authors.

credit_email_biotools Like *credit_name_biotools*, but for the email attribute.

corresp_author_phone Like *corresp_author_name*, but for telephone numbers of corresponding authors.

corresp_author_uri Like *corresp_author_name*, but for web pages of corresponding authors.

credit_url_biotools Like *credit_name_biotools*, but for the URL attribute.

credit The *credit is constructed* in the second pass (*-pass2*) from the corresponding authors of publications (with possible duplicates being merged). The name, ORCID iD, e-mail and URL can be filled, with only non-empty values output to the column and separated by `", "` and values of different credits separated by `" | "`. The value of this column is suggested as the content of the credit attribute of the tool for bio.tools.

3.3 diff.csv columns

biotools_id The first column lists the bio.tools ID of an existing bio.tools entry the current row of suggestions is about. If a new entry constructed by Pub2Tools is determined to be *existing in bio.tools*, then it will not be output to *to_biotools.json*, but instead redirected here. Values of both the new entry and the entry existing in bio.tools are output to *results.csv* and the corresponding row there can be found be searching for the ID present here in the column *existing* of `results.csv`.

However, if no differences are found between the new entry and the entry existing in bio.tools (and *possibly_related* is also empty), then nothing is output also to `diff.csv`. To be more precise, by differences we mean clashes between values of the new entry and the bio.tools entry or values which exist only in the new entry – so values that exist in the bio.tools entry and not in the new entry constructed by Pub2Tools are not considered to be different and nothing is suggested about them.

score_score2 A combined score (either equal to *score2* or to *score* + 10000 in case *score2* is not calculated) of a new entry constructed by Pub2Tools, which more or less shows the confidence that the correct tool name was extracted from the publication(s) in the new entry. Entries of the *diff.csv* spreadsheet are sorted by this score, unless there are multiple entries with the same *biotools_id*, in which case these entries are grouped together next to the highest scored such entry (this can happen for example when a bio.tools entry has multiple publications and distinct new Pub2Tools entries each match one of these publications).

current_publications The publication IDs (separated by " | ") of the existing bio.tools entry. The value in this column is only filled if any of the columns *modify_publications*, *add_publications* or *modify_name* contain some non-empty value.

modify_publications Contains publication IDs of the new entry constructed by Pub2Tools that have a conflict with some existing publication IDs of the current bio.tools entry. A conflict means that there is a match between some members of the publication ID triplets [PMID, PMCID, DOI] of the entries, but some other non-empty members are not equal. This indicates a mistake either in bio.tools (which happened for example when manually entering a publication ID) or in the entry constructed by Pub2Tools (where publication information came from an external service, like Europe PMC). So publication IDs here could be compared to the corresponding publication IDs in *current_publications* and by checking the publication online it can be decided which one is correct and if modifications have to be made in bio.tools.

Note: In principle, this column could also contain cases, where some existing publication ID has some empty parts (PMID, PMCID or DOI), which could be filled by information found by Pub2Tools, however such cases are not output here as such filling could be done automatically without any need for curation (see https://github.com/bio-tools/biotoolsLint/issues/2#issuecomment-427509431).

---

add_publications Contains publication IDs (separated by " | ") of the new entry constructed by Pub2Tools that are missing in the matched existing entry currently in bio.tools. Thus, the publication IDs listed here could be added to the existing bio.tools entry. However, sometimes the suggestion in this column is wrong (for example, when *suggestions were merged* incorrectly in Pub2Tools because the names of distinct tools were exactly equal), but sometimes a value here could also indicate mistakes in bio.tools (like an incorrect publication attached to a tool or the same tool duplicated in bio.tools, but with different publications).

current_name The name of the existing bio.tools entry. The value in this column is only filled if *modify_name* contains some non-empty value, that is, if it is suggested to change the name currently in bio.tools.

modify_name Contains the *name suggestion* of the new entry constructed by Pub2Tools if it differs from the name currently existing in bio.tools (output to *current_name*). Whether the name should actually be modified in bio.tools, is up to the curator.

In many cases, both *current_name* and *modify_name* list quite obviously the same tool name, but with a slight difference in capitalisation, punctuation, whitespace, version number being present, name being an acronym, etc. And these small differences can matter, for example the tools coMET (1), Comet (2), CoMet (3) or PRISM (1), PriSM (2), PrISM (3) are all distinct tools with the only difference in the names being the capitalisation.

Note: Pub2Tools doesn't really take into account the Curators Guide's rules for the name attribute, thus in some cases the value in *current_name* will actually be correct.

In some cases, very different names are listed by *current_name* and *modify_name*. This can happen, if a wrong publication is attached to a tool in bio.tools, if Pub2Tools failed to extract the correct name, if a bio.tools entry is a conglomeration of differently named subtools, if a very general publication is attached to a more specific constituent subtool, if an attached publication is only indirectly related to the tool, etc.

The lower in the table, the more probable it is, that Pub2Tools failed to extract the correct name, thus for entries with "very low" *confidence* (*score_score2* is less than 1072.1) the columns *current_name* and *modify_name* will be empty even if there are differences in names.

possibly_related Contains bio.tools IDs (separated by " | ", with each ID followed by the name in parenthesis) of existing entries of bio.tools that might be related to the new entry constructed by Pub2Tools. It lists entries where evidence was not enough to say that the new entry is a duplicate of the listed entries. This happens, when names were matched (*name_existing_publication_different* or *name_match*), but no publications, links or credits could additionally be matched, or when solely some links could be matched (*link_match*). As such, this

column contains mostly unrelated entries, however, sometimes the entries could actually be related and require some curation decisions (removal, combining of entries, etc).

current_homepage The homepage of the existing bio.tools entry (also output to *homepage_biotools* of `results.csv`). Not filled, if *modify_homepage* is empty. If the homepage is determined to be broken in bio.tools, then `(homepage_status: 1)` will follow the URL. If it is determined to be broken by Pub2Tools, then `(broken)` will follow.

modify_homepage The new *homepage* as suggested by Pub2Tools. A new homepage is suggested as replacement for *current_homepage* if the homepage of the new entry constructed by Pub2Tools does not match the homepage of the existing bio.tools entry and one of the following holds: *current_homepage* is broken (according to both bio.tools and Pub2Tools) or the URL of the new homepage is determined to be a *link* with type "Other". Note, that the new and existing homepages are also considered equal if they redirect to the same final URL, also, `www`, `index.html`, etc are ignored and comparison of the domain name part is done case-insensitively.

If *current_homepage* is suggested to be replaced, then Pub2Tools might add the URL in *current_homepage* to *add_links*, *add_downloads* or *add_documentations*, that is, the homepage of the existing bio.tools entry should not simply be thrown away but added to some other bio.tools link attribute. If *current_homepage* is not suggested to be replaced, the this column would be empty and Pub2Tools might instead add the homepage of the new entry to *add_links*, *add_downloads* or *add_documentations*.

The URL suggested as the new homepage has the limitation that it must have occurred somewhere in a publication abstract or full text. Which means, that the URL in *current_homepage* might actually be a better homepage that just doesn't occur in the publication text. It's up to the curator to decide whether to perform the replacement – and if the replacement is not done, then the new homepage should not simply be thrown away, but considered for addition to *link*, *download* or *documentation* beforehand. The new homepage extracted by Pub2Tools could also be plainly incorrect and the probability of this increases the further down the entries we move. So, if *confidence* is "very low" (*score_score2* is less than 1072.1), then the new homepage is always thrown away and *current_homepage* and *modify_homepage* will always be empty.

current_links URLs currently in the link attribute of the existing bio.tools entry (also output to *link_biotools* of `results.csv`). Links are separated by `" | "` and each URL is followed by the link type in parenthesis. Not filled, if no new links to add are present in the entry constructed by Pub2Tools (that is, *add_links* is empty) or if there are simply no link attribute links currently in the existing bio.tools entry.

add_links URLs from *link* of the new entry constructed by Pub2Tools that are missing in the currently existing entry of bio.tools and thus could be added there. Links are separated by `" | "` and each URL is followed by the link type in parenthesis. Sometimes, a link could be incorrectly categorised, as whether it should go to *link*, *download* or *documentation* is based solely on the URL string. Also, if *confidence* is "very low" (*score_score2* is less than 1072.1), then confidence in the correctness of the new links found by Pub2Tools is too low and thus these new links will be thrown away and *current_links* and *add_links* will by empty.

current_downloads Like *current_links*, but concerning the download attribute and *download_biotools*.

add_downloads Like *add_links*, but concerning *download* and adding to *current_downloads*.

current_documentations Like *current_links*, but concerning the documentation attribute and *documentation_biotools*.

add_documentations Like *add_links*, but concerning *documentation* and adding to *current_documentations*.

current_license The license currently set as the value of the license attribute of the existing bio.tools entry (also output to *license_biotools* of `results.csv`). Not filled, if *modify_license* is empty, that is, no licenses were extracted by Pub2Tools for the new entry or the found license is equal to the license in the existing bio.tools entry.

modify_license The *license* of the new entry constructed by Pub2Tools that should replace the (either different or missing) license information of the existing bio.tools entry displayed in *current_license*. New license information is extracted from web pages (mostly repositories, like GitHub and Bioconductor) and publication abstracts, which means we can add provenance information, that is web page URLs and publication IDs (separated by `",`

"), after the license string in parenthesis. If *confidence* is "very low" (*score_score2* is less than 1072.1), then confidence in the correctness of the extracted tool name and thus in the correctness of the extracted web pages is too low, so in that case only license information extracted from publication abstracts is considered (that is, *license_abstract* is used instead of *license*).

current_languages The languages (separated by " | ") currently set as the value of the language attribute of the existing bio.tools entry (also output to *language_biotools* of `results.csv`). Not filled, if *add_languages* is empty, that is, no languages were extracted by Pub2Tools for the new entry or all found languages are already present in the existing bio.tools entry.

add_languages A list of *language* strings (separated by " | ") from the new entry constructed by Pub2Tools that are different from the languages in the existing bio.tools entry (displayed in *current_languages*) and thus should be added there. New language information is extracted from web pages (mostly repositories, like GitHub and Bioconductor) and publication abstracts, which means we can add provenance information, that is web page URLs and publication IDs (separated by ", "), after the each language string in parenthesis. If *confidence* is "very low" (*score_score2* is less than 1072.1), then confidence in the correctness of the extracted tool name and thus in the correctness of the extracted web pages is too low, so in that case only language information extracted from publication abstracts is considered (that is, *language_abstract* is used instead of *language*).

current_credits The credit information currently set as the value of the credit attribute of the existing bio.tools entry (also output to *credit_name_biotools*, *credit_orcidid_biotools*, *credit_email_biotools* and *credit_url_biotools* of `results.csv`). The credit entries are separated by " | " with each entry in the form `name, ORCID iD, e-mail, URL`, where any missing attribute is simply omitted. Not filled, if *modify_credits* and *add_credits* are empty.

modify_credits *Credit* entries from the new entry constructed by Pub2Tools that have a match with an existing credit in *current_credits* through the name, ORCID iD or e-email (a match does not mean equality, for example a person's name can be written with an academic title and abbreviated middle name, while omitting accents), but where the new credit has information missing in the existing credit or there are slight differences in the name, ORCID iD or e-mail. Whether the missing information or the slight variations are important, is left to decide by the curator.

add_credits *Credit* entries from the new entry constructed by Pub2Tools that are missing in the existing bio.tools entry (displayed in *current_credits*) and thus could possibly be added to the existing entry. Credits are displayed as in *current_credits*: separated by " | " with each credit in the form `name, ORCID iD, e-mail, URL`, where any missing attribute is simply omitted. One possible caveat: if bio.tools contains only a person's e-mail and Pub2Tools extracts only the name of the same person, then these cannot be automatically connected currently and the name is added here instead of the correct column *modify_credits*.

3.4 to_biotools.json attributes

The final results file *to_biotools.json* will contain entries where *include* is `true` and *existing* is empty. It is a JSON file containing a number (named "count") specifying how many entries there are and an array (named "list") containing each entry as a JSON object with the following structure:

name The name of the tool from *suggestion*. The name is not necessarily unique within a JSON file – equal names are indeed merged into one entry, but this is not done for entries with a "very low" *confidence*. Generating a unique bio.tools ID is also not done, this is left to the importer of the JSON file.

description The description candidates and messages to the curator from *description*.

homepage The homepage of the tool from *homepage*.

function[] The function attribute is an array containing EDAM operations (but also EDAM data and format) found by the *-map* step. Pub2Tools outputs all found EDAM operations under one function (see tool functions), so the size of the array is always 1 when any EDAM operations are found.

operation[] An array containing the found EDAM operation terms.

uri The URI of the EDAM term.

term The label of the EDAM term.

note The *-map* step can also propose candidate EDAM terms from the data and format branches (if requested), however, these will need to be divided into the input object and output object and EDAMmap can't differentiate between inputs and outputs. Thus, EDAM data and format terms will be output under `note` as a string with the following format: `EDAM_URI (EDAM_label) | EDAM_URI (EDAM_label) | ...`.

topic[] The topic attribute is an array containing the EDAM topic terms found by the *-map* step.

uri The URI of the EDAM term.

term The label of the EDAM term.

language[] An array containing the strings of all languages of the tool from *language*. Unfortunately, biotoolsSchema does not leave space for outputting the web page URLs and publication IDs where these languages where found from, so if this extra information seems important for making curation decisions, then it can be looked up from the *language* column of *results.csv*.

license The license of the tool from *license*. Unfortunately, biotoolsSchema does not leave space for outputting the web page URLs and publication IDs where the license was found from, so if this extra information seems important for making curation decisions, then it can be looked up from the *license* column of *results.csv*.

link[] An array of miscellaneous links of the tool from *link*.

url The URL of the link.

type[] The link type; an array with exactly one element as currently Pub2Tools only finds exactly one type for each link.

download[] An array of download links of the tool from *download*.

url The URL of the link.

type[] The download type; an array with exactly one element as currently Pub2Tools only finds exactly one type for each download link.

documentation[] An array of documentation links of the tool from *documentation*.

url The URL of the link.

type[] The documentation type; an array with exactly one element as currently Pub2Tools only finds exactly one type for each documentation link.

publication[] The publication attribute is an array filled with publications where the tool was extracted from. Normally, one publication can produce one tool entry for bio.tools, but sometimes multiple tool suggestions can be *merged into one result*, thus the size of the array can be greater than 1.

doi The DOI of a publication from *doi*.

pmid The PMID of a publication from *pmid*.

pmcid The PMCID of a publication from *pmcid*.

credit[] An array of credits of the tool from *credit*.

name The name of a credit.

email The e-mail of a credit.

url The URL of a credit.

orcidid The ORCID iD of a credit.

typeEntity The entity type of a credit. Always "Person", because currently the only source for credits is the corresponding authors of publications.

confidence_flag From *confidence*, so either "high", "medium", "low" or "very low".

Note: Empty or null values will be omitted from the output.

As an example, consider the following new entry:


```
{
  "name" : "PAWER",
  "description" : "Protein Array Web ExploreR.\n\npaweR is an R package for analysing␣
↪protein microarray data.\n\nWeb interface for PAWER tool (https://biit.cs.ut.ee/
↪pawer/).",
  "homepage" : "https://biit.cs.ut.ee/pawer",
  "function" : [ {
    "operation" : [ {
      "uri" : "http://edamontology.org/operation_3435",
      "term" : "Standardisation and normalisation"
    }, {
      "uri" : "http://edamontology.org/operation_0337",
      "term" : "Visualisation"
    }, {
      "uri" : "http://edamontology.org/operation_2436",
      "term" : "Gene-set enrichment analysis"
    } ],
    "note" : "http://edamontology.org/data_2603 (Expression data) | http://
↪edamontology.org/data_2082 (Matrix) | http://edamontology.org/data_0958 (Tool␣
↪metadata) | http://edamontology.org/data_3932 (Q-value) | http://edamontology.org/
↪format_3829 (GPR) | http://edamontology.org/format_1208 (protein) | http://
↪edamontology.org/format_3752 (CSV)"
  } ],
  "topic" : [ {
    "uri" : "http://edamontology.org/topic_3518",
    "term" : "Microarray experiment"
  }, {
    "uri" : "http://edamontology.org/topic_0121",
    "term" : "Proteomics"
  }, {
    "uri" : "http://edamontology.org/topic_0203",
    "term" : "Gene expression"
  }, {
    "uri" : "http://edamontology.org/topic_0769",
    "term" : "Workflows"
  }, {
    "uri" : "http://edamontology.org/topic_0632",
    "term" : "Probes and primers"
  } ],
  "language" : [ "R" ],
  "link" : [ {
    "url" : "https://gl.cs.ut.ee/biit/paweR",
    "type" : [ "Other" ]
  }, {
    "url" : "https://gl.cs.ut.ee/biit/pawer_web_client",
    "type" : [ "Other" ]
  } ],
  "publication" : [ {
```


(continues on next page)

(continued from previous page)


```
    "doi" : "10.1101/692905"
  }, {
    "doi" : "10.1186/S12859-020-03722-Z",
    "pmid" : "32942983",
    "pmcid" : "PMC7499988"
  } ],
  "confidence_flag" : "high"
}
```


The example is missing the following fields: `license`, because license information could not be extracted from the publication abstract and there were also no (usually repository) links where this information could be found; `credit`, which has been manually removed from the example; `download` and `documentation`, as none of the links matched to the name of the tool and extracted from the publication abstract and full text are categorised as such.

3.5 Performance

On the 6th of August 2019, Pub2Tools was run for the months of May, June and July 2019. The results can give a rough estimate of its performance.

Extracting new tools from 1 month worth of publications took Pub2Tools about 1h 40min (1h 15min of it was spent on downloading the publications) with default parameters.

The total number of publications returned from Europe PMC for `CREATION_DATE:[2019-05-01 TO 2019-05-31]` was around 123000, for June the number was 115000 and for July 111000. After prefiltering with the *-select-pub* step (with default options), these numbers were reduced to 2429, 2365 and 2253 for May, June and July respectively. So, such prefiltering allows to reduce the number of publications to be fetched to around 2% of the initial availability (of course, the cost is that a few valid publications will also be thrown away). After running all the steps of Pub2Tools, the number of entries written to *to_biotools.json* were 689, 670 and 670 for May, June and July respectively. A manual inspection of the results revealed, that around 20% of the entries were not publications about tools, databases or services and thus were unsuitable for bio.tools (but even for "very low" *confidence* entries, roughly half seemed to be about a tool, though the name was quite often wrongly extracted). So, in the year 2019, roughly 500 new entries per month could be added to bio.tools, which is a bit less than 0.5% of all articles available through PubMed. In addition to the new entries, some results were found to be already existing in bio.tools: the file *diff.csv* contained 82, 37, 29 entries for May, June, July.

On the 1st of October 2020, Pub2Tools was run for the months of August and September 2020. Extracting new tools from 1 month worth of publications took Pub2Tools about 2h with default parameters (around 1h of it was spent on downloading publications and 30min on downloading web pages).

The following table shows the percentage of potential new entries whose attribute was filled with at least some value per each attribute:

attribute	2019-05	2019-06	2019-07	2020-08	2020-09
pmid	75.04%	71.64%	68.21%	68.55%	61.99%
pmcid	43.11%	39.10%	40.60%	35.36%	28.92%
doi	99.85%	99.55%	99.40%	99.67%	98.69%
homepage	80.41%	80.00%	84.18%	85.64%	83.79%
link	17.42%	17.31%	16.57%	18.82%	17.74%
download	1.89%	1.19%	2.84%	3.26%	3.07%
documentation	3.77%	3.13%	4.33%	4.79%	3.50%
license	22.35%	25.52%	25.82%	23.39%	23.77%
language	47.90%	52.69%	54.03%	50.27%	49.73%
credit	42.09%	35.67%	37.91%	72.58%	70.65%

The corresponding figure:

The name and publication are always filled, because all entries are extracted from some publication and a name has to be extracted and chosen. The description is also always filled, though it always requires curation also and in case of missing links will contain only text from the publication abstract. The homepage is also a required attribute, however it will be reported unfilled here in case a homepage could not be found and the homepage attribute was just filled with a link to the publication.

The falling fill rate of PMID and PMCID points to the growing share of pre-prints (findable through Europe PMC). For 2020-08 and 2020-09 the fill rate of credit information is a lot higher because in January 2020 support was added for getting corresponding author information directly from web pages of articles of many journals (resolved through the publication DOI) in addition to getting corresponding authors information from PubMed Central (i.e. only for publications that have a PMCID).

Note: Pub2Tools sometimes also extracts and writes incorrect information to an attribute (except publication and credit information which is mostly correct), so the percentages presented in the table would be slightly lower if only correctly filled attributes would be taken into account. On the other hand, if only high confidence entries would be taken into account, then the fill rates of homepage, link, license, language and credit would be roughly 10 percentage points higher.

CHAPTER 4

Ideas for future

Sometimes ideas are emerging. These are written down here for future reference. A written down idea is not necessarily a good idea, thus not all points here should be implemented.

- Fix potential systematic issues discovered from feedback of curation of Pub2Tools results.
- Currently, Pub2Tools has been tweaked in the direction of having less false positives (for example, even roughly half of the entries with "very low" *confidence* are about tools) to not discourage curators too much into looking at all results of Pub2Tools. But maybe the amount of false negatives could also be decreased further without too much cost, for example by *running Pub2Tools on existing bio.tools content* and checking if there are any ways to reduce entries with *include* `false`.
- Also, if some entries are added through some other means than Pub2Tools, then Pub2Tools could be run on publications of these entries and then it could be checked how many of these entries Pub2Tools fails to include and why.
- The default values of the mapping parameters in the *-map* step are equal to the default values defined for the parameters in EDAMmap. Maybe some defaults could be changed for Pub2Tools, for example the number of terms output per branch?
- Maybe the *score* or *score2* could also be modified by how well the entry could be mapped to EDAM terms by EDAMmap in the *-map* step?
- Could the currently unused *citations_count_normalised* also be used for modifying the goodness of entries? Note, that its value could only be useful for older publications as these have had enough time to be cited.
- Could MeSH terms be used in *-select-pub* to prefilter the publications in addition to or instead of selection based on occurrence of phrases in the abstract and title? PubFetcher and the mesh field of its database can be used to look for prevalence of MeSH terms in publications currently in bio.tools. Note, that the selection by MeSH terms can't return recent publications, as these terms are not added to articles right away (for example, it might take half a year after the publication date for the terms to appear).
- How to best present the provenance (i.e. publication IDs and web page URLs where information was found) of the licenses and languages to the curator? As this extra information can't be embedded into *to_biotools.json* and it must currently manually be looked up from *results.csv*.

- Currently, the *merging of supposedly duplicate results* is done base on the exact equality of the *suggested names* only. Maybe further features should be taken into account, like matching links or credits, to reduce the number of wrongly merged results?
- Further links could potentially be found by looking for links in the content of web pages extracted from the publication abstract and fulltext. But to avoid false positive, maybe links from the same domain should only be considered and maybe "about", "help", etc should be looked for in the URL string? Also, in some cases further links could be automatically inferred, like a documentation URL for a given CRAN URL.
- Currently, each link receives exactly one type (like "Repository" or "Mailing list"), however biotoolsSchema allows for more than one type to be specified.
- Explore the possibilities of extracting tools through other means than publications present in Europe PMC. Maybe from repositories, like GitHub, CRAN, Zenodo, etc?
- Try running Pub2Tools on all publications from a given period, i.e. without prefiltering with *-select-pub*. This could be achieved for the Open Access subset of articles, that can be downloaded in bulk (as using PubFetcher to download millions of articles would be too wasteful and slow).
- Try to automatically fill further attributes, without causing too many false positives, for example operating system or the bio.tools specific tool type.
- Currently, the *description* is filled with candidate phrases that the curator must choose from or combine. Automatic text summarisation could be tried to automatically construct the final description proposal. Or, just choose one/two of the candidate phrases automatically (instead of the curator making the choice).
- Try to figure out which of the description *messages* are actually useful.
- The *credit* is currently filled only from corresponding authors of publications. Explore other possibilities to find credit information, for example contact information is sometimes mentioned in the publication abstract.