Design Automation Renegades

Globetrotting Division

BibTeX Analytics: For Automating Reference Management and Recognizing Emerging Trends

Zhiyang Ong [1]

A document on *Python*-based BibTeX analytics
For Reference Management ...
and Emerging Trend Recognition

June 7, 2018

[1]Email correspondence to: ongz@acm.org

Abstract

This documents how the repository of the BibTeX *Analytics* project is organized, and its software architecture. It also describes the future goals of the project for using a data analytics approach to recognize emerging trends in research, especially emerging research trends in electrical and computer engineering, computer science, and other fields, such as medicine, agriculture, and environmental science.

Insert abstract here.

More stuff to be included.

Revision History

Revision History:

1. Version 0.1, May 21, 2018. Initial copy of the report.
2. Version 0.2, May 23, 2018. Updated all chapters of the report.

Contents

Chapter 1

Organization of the BibTeX Analytics Repository

The main deliverables of the BibTeX *Analytics* project are a *Python*-based software to perform reference management, and data analytics on BibTeX entries to recognize emerging research trends.

The organization of the BibTeX *Analytics* repository is described as follows:

1. analysis:
   (a) Scripts to analyze BibTeX databases:
      i. To remove metadata from them.
      ii. To determine the set of keywords/keyphrases from their BibTeX entries.
      iii. To determine the set of publishers from their BibTeX entries.
      iv. To determine the set of series from their BibTeX entries.
      v. To determine the set of journals from their BibTeX entries.
      vi. To determine the set of authors from their BibTeX entries.
      vii. To determine the set of year from their BibTeX entries.
2. analytics_engine:
   (a) classification:
      i. Classify each author into hot or not.
      ii. Classify each keyword/keyphrase (and sets thereof) into hot or not.
   (b) clustering:
      i. Find clusters of keyphrases from sets of keyphrases, where each set is a set of keyphrases for a publication, and determine the top 5/10 most common subsets/clusters of keyphrases.
   (c) prediction:
      i. Use predictive analytics to predict emerging trends in research.
      ii. Use predictive analytics to predict emerging megatrends, or macrotrends.
3. `automated_regression_testing.py`:
   (a) Run as: **./automated_regression_testing.py**
   (b) No input nor output required.
   (c) *Python* script to automate regression testing.
   (d) **Sort of deprecated.** Still works though.
4. big_input:
   (a) Data set for stress testing the software deliverable of the BibTeX *Analytics* project.
5. database

   (a) *bibtex_database* (`bibtex_database.py`) class represents (each instance of) a BibTeX database of BibTeX entries.
   (b) `bibtex_database_test.py` is a *Python* script to test the functionality of the *bibtex_database* class.
   (c) *entry* (`entry.py`) class represents each (instance of a) BibTeX entry.
   (d) `entry_test.py` is a *Python* script to test the functionality of the *entry* class.
   (e) `key_check.py` is a *Python* script to check if each BibTeX entry is valid.
   (f) `key_check_test.py` is a *Python* script to test the functionality of the *Python* script `key_check.py`.
   (g) `key_frequency_pairs.py` is a *Python* script to demonstrate how to sort a set of 2-tuples based on its first/former field and its second/last/latter field.

6. `duplicate_BibTeX_entries.py`:
   (a) Run as: **./duplicate_BibTeX_entries.py [-h] [BibTeX file]**
   (b) A *Python* script to determine if duplicate BibTeX entries exist in a BibTeX file/database. If such entries exist, warn the user that duplicate BibTeX entries exist.

7. `editions.py`:
   (a) Run as: **./editions.py [input BibTeX file] [-h]**
   (b) A *Python* script to display a set of editions from all the BibTeX entries in a BibTeX file/-database.

8. front_end:
   (a) Parse input BibTeX files, create BibTeX entries (i.e., instances of *entry*), and insert them into the BibTeX database, an instance of *bibtex_database.py*.

9. `incremental_test.py`:
   (a) Run as: **./incremental_test.py [input BibTeX file]**
   (b) A *Python* script to incrementally test features for performing reference management and data analytics operations with BibTeX files/databases.

10. input:
   (a) A set of BibTeX files to test my *Python*-based BibTeX *Analytics* software.

11. `institutions.py`:
   (a) Run as: **./institutions.py [input BibTeX file] [-h]**
   (b) A *Python* script to display a set of institutions from BibTeX entries in a BibTeX file/database.

12. `journal_titles.py`:
   (a) Run as: **./journal_titles.py [input BibTeX file] [-h]**
   (b) A *Python* script to display a set of journal titles from BibTeX entries in a BibTeX database.

13. `keywords_display.py`:
   (a) Run as: **./keywords_display.py [input BibTeX file] [-h]**
   (b) A *Python* script to display a set of keywords/keyphrases from BibTeX entries in a BibTeX database.

14. makefile:
   (a) For build automation of *Python* scripts, not placed in subdirectories, in the repository.

15. notes:
   (a) `gpl-license.text`, `LICENSE`, and `mit-license.text` are text files of the GNU General Public License (GNU GPL) (`gpl-license.text`) and The MIT License (`LICENSE` and `mit-license.text`)
   (b) guidelines:

i. A document containing a set of guidelines on how to collaborate with me.

(c) report:

i. This document that describes the organization of the BIBTEX *Analytics* repository, the software architecture of the BIBTEX *Analytics* software, and future work of the BIBTEX *Analytics* project.

16. `organizations.py`:
    (a) Run as: **./organizations.py [input BibTeX file] [-h]**
    (b) A *Python* script to display a set of organizations from BIBTEX entries in a BIBTEX database.
17. `publishers.py`:
    (a) Run as: **./publishers.py [input BibTeX file] [-h]**
    (b) A *Python* script to display a set of publishers from BIBTEX entries in a BIBTEX database.
18. `readme.md`:
    (a) A *Markdown*-based `readme` document briefly describing this project.
19. `rm_bibtex_metadata.py`:
    (a) Run as: **./rm_bibtex_metadata.py [input BibTeX file] [output BibTeX file] [-h]**
    (b) *[output BibTeX file]* is an optional parameter.
    (c) A *Python* script to delint/remove BIBTEX metadata from a BIBTEX database/file.
20. sandbox:
    (a) A set of *Python* scripts to test different concepts in *Python*.
21. statistics:
    (a) `test_statistics_tester.py` is a *Python* script to test the functionality of the *test_statistics* class.
    (b) *test_statistics* (`test_statistics.py`) class to perform statistical analysis on results of automated testing of a *Python* script.
22. `tutti_series.py`:
    (a) Run as: **./tutti_series.py [input BibTeX file] [-h]**
    (b) A *Python* script to display series from BIBTEX entries in a BIBTEX database.
23. utilities:
    (a) `file_io.py` is a *Python* script to perform input/output (I/O) operations on files, such as BIBTEX databases/files and LATEXdocuments.
    (b) `queue_ip_arguments.py` is a *Python* script to process input arguments for a script to clean BIBTEX databases/files.
24. `validate_url.py`:
    (a) Run as: **./validate_url.py [input BibTeX file] [output BibTeX file] [-h]**
    (b) *[output BibTeX file]* is an optional parameter.
    (c) A *Python* script to check each BibTeX entry in a BibTeX database if it has the non-standard BIBTEX field(s) "Bdsk-Url-1" (and "Bdsk-Url-2"), and if the "Url" (and "Doi") field(s) is(/are) missing; if these conditions are true, copy their values to the "Url" BibTeX field (and "Doi" field, if it is a DOI).
25. `z_booktitles.py`:
    (a) Run as: **./z_booktitles.py [input BibTeX file] [-h]**
    (b) A *Python* script to display booktitles from all BIBTEX entries in a BIBTEX database.

Chapter 2

Software Architecture of the BibTeX Analytics Project

Figure 2.1 shows the software architecture of the BibTeX *Analytics* project. It shows the different packages of the BibTeX *Analytics* project, as well as the classes in each package.

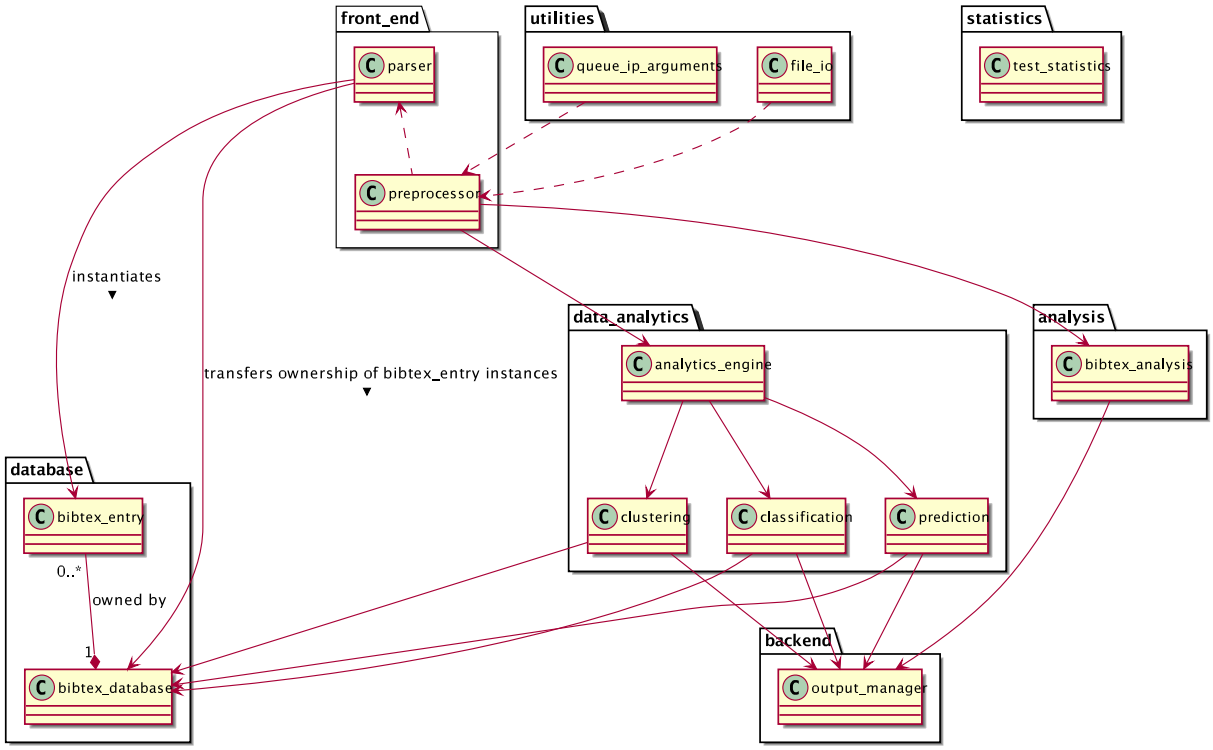

Figure 2.1: Software architecture of the BibTeX *Analytics* project.

The `front_end` package performs the following operations: interacts with the `utilities` package to process command-line input arguments from the user and parse a BibTeX file, instantiates a `bibtex_database` object to contain instances of `bibtex_entry` objects, instantiates a `bibtex_entry` object for each BibTeX entry in the BibTeX file, and passes `bibtex_entry` objects to the `bibtex_database` object for storage. Hence, this package also interacts with the `database` package. Using input from the command-line input arguments, it either calls static methods of classes in the `data_analytics` package

or the `analysis` package.

When objects in the `data_analytics` package or the `analysis` package performs operations specified by the command-line input arguments, it interacts with the `bibtex_database` object in the `database` package to access required information to perform the aforementioned operations. When these operations are completed, objects from either of these packages, `data_analytics` or `analysis`, pass the computation results to the `output_manager` in the `backend` package. The `output_manager` either prints the computation results to a text file in the current working directory or to standard output for display on the *Terminal* application.

In the `utilities` package, the `queue_ip_argument` class has static methods to process command-line input arguments from the user. Likewise, the `file_io` class has static methods to process an input BIBTEX file, which is specified by the user in the command line.

Regarding the `data_analytics` package, the `analytics_engine` class manages the operations in data analytics, and calls the appropriate machine learning tool. The machine learning tools are: `clustering`, `classification`, and `prediction` (for predictive analytics).

Lastly, Figure 2.2 shows the software architecture of the BIBTEX *Analytics* project that includes the highly coupled `test_statistics` class. The `test_statistics` class collects data from running software test suites during automated regression testing, performs simple statistical analysis on the test data, and reports the result to the user.

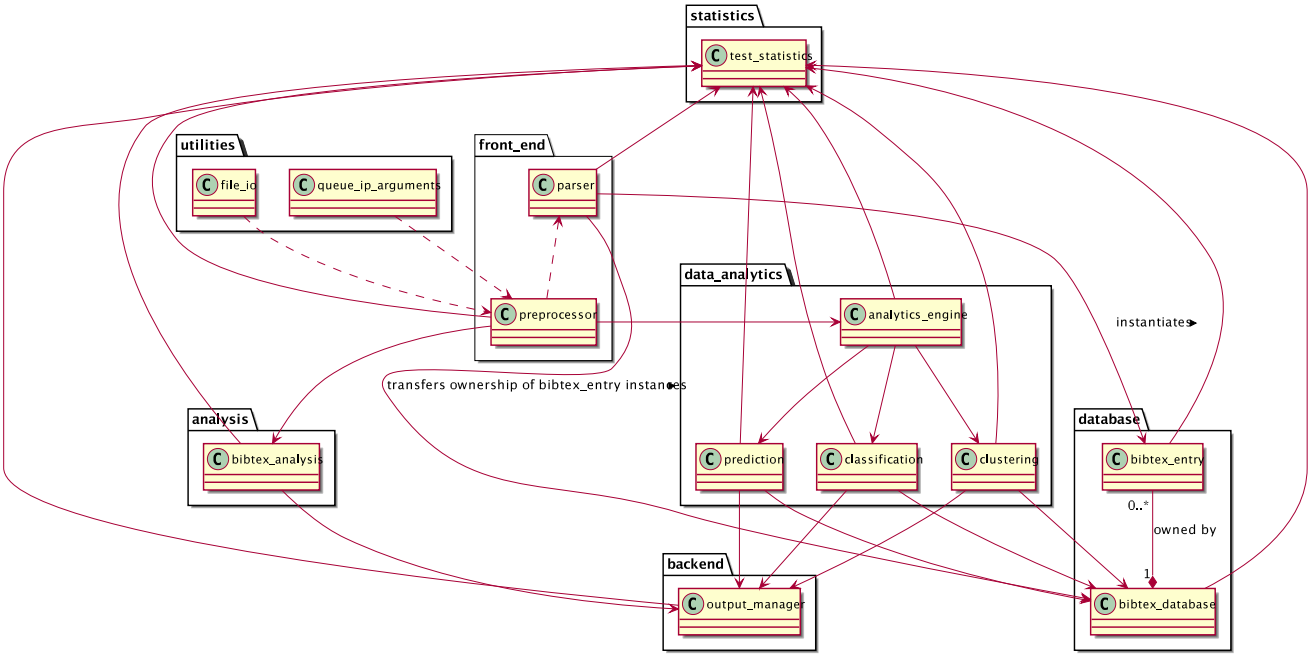

Figure 2.2: Software architecture of the BIBTEX *Analytics* project, with highly coupled `test_statistics` class.

Chapter 3

Future Work

Future work of the BibTeX *Analytics* project is described as follows:

1. Clustering of keywords/keyphrases:
   (a) **Problem statements**:
      i. For an author, find clusters of keyphrases, publishers, journal titles, conferences, ...
      ii. For each keyphrase, determine the cluster of publishers, years, journal titles, conferences, ...
   (b) Build dictionary of *(keyphrase, frequency)* two-tuples (or pairs).
   (c) Sort the dictionary based on the frequency term/element, *frequency*, in these two-tuples.
   (d) Alternate solution:
      i. Build a set of *(keyphrase, [list of years])* lists.
      ii. *[list of years]* is a list of years of publications; or it is a set of years for publications that include the keyphrase *keyphrase* in its set of keyphrases.
      iii. Sort the set based on length of the list of years, *[list of years]*.
   (e) If possible, visualize the data for this.
   (f) Alternate solution:
      i. For each BibTeX entry, there exists a set/cluster of keyphrases.
      ii. Find overlaps/intersections between these sets.
      iii. E.g., for each set of keyphrases (associated with a publication) with at least three (i.e., more than two) terms, build a list of non-empty overlaps.
      iv. Find the largest intersecting subset among the non-disjoint sets. Or, find the top 5-10 most common overlaps.
      v. Refer to books on data visualization for information on visualizing this.
      vi. Also, refer to this technical report from Stanford, "Union-member algorithms for non-disjoint sets." See https://dl.acm.org/citation.cfm?id=892212.
   (g) Compare problem with common subexpression elimination in compiler design, and maximum clique covering problem,
   (h) The more common/frequent the subsets are, the hotter the subsets are.
   (i) Therefore, find the most frequent intersection. And/or, the greatest intersection.
   (j) Problem restated: For each keyphrase, find the largest intersection it has with all the other sets containing the keyword.
   (k) That is, capture the largest intersection(s) and map it(/them). This is because multiple sets of the same size could exists. Note that the $2^{nd}/3^{rd}$ largest intersections includes the largest intersection(s) minus 1 (or 2) term(s).
   (l) Find overlaps/intersections between these overlaps/intersections.
2. Classification:

   (a) Classify each keyphrase/topic into hot or not.
   (b) Classify each author into hot or not.
   (c) Note that since the size of my BibTeX database is small, but significant, compared to reality, put these results of classification into proper perspective.
3. Predictive analytics:
   (a) Recognize trends, predict trends.
   (b) For the last 5 (or 8) years, find the (emerging) trends of hot topics.
4. For a given keyphrase, provide a list of BibTeX entries that contain that keyphrase in their keyword BibTeX field.
5. Perform miscellaneous tasks to clean up the BibTeX file.
6. Check if the ampersand is surrounded by curly braces and set to the normal (non-Italics) font.
7. For each conference proceedings, check if its abbreviation is placed within round brackets after the title of the conference proceedings. Also, Check if there is no comma between the title and the abbreviation.
8. Write a script to extract the keywords from the BibTeX repository, arrange them in alphabetical order, and pipe them to an output file.
9. Check if the addresses of the publications have the U.S. states in capital letters.:
   (a) If I use abbreviations for states and territories in Australia and Canada, do likewise.
   (b) For publications outside the U.S., (and Australia and Canada), ignore this.
10. Check if DOIs and/or URL fields are missing, if the following fields (metadata for *BibDesk*) exists:

    (a) Bdsk-Url-1
11. Additional tasks:
    (a) `extract_citations.py`:
        i. Run as: *./extract_citations.py [LaTeX sources] [BibTeX output]*
        ii. Produces an intermediate output, which is a set of BibTeX keys that uniquely identifies a matching BibTeX entry in my BibTeX database.
    (b) `not_defined_references.py`:
        i. Run as: **./not_defined_references.py [LaTeX sources] [BibTeX input]**
        ii. Check if this citation uses a undefined reference
        iii. No output required.
    (c) `uncomment_latex_src_files.py`:
        i. Run as: **./uncomment_latex_src_files.py [dirty LaTeX source files] [clean LaTeX source files]**
        ii. Remove comments from LaTeX source files. Non importante.
12. Find emerging research trends to consider pivoting towards, or to get involved in side projects
    (a) E.g., benchmark adiabatic quantum computers with topological computers and universal quantum computers [1].

Bibliography

[1] Prateek Tandon, Stanley Lam, Ben Shih, Tanay Mehta, Alex Mitev, and Zhiyang Ong. Quantum Robotics: A Primer on Current Science and Future Perspectives, volume 10 of Synthesis Lectures on Quantum Computing. Morgan & Claypool Publishers, San Rafael, CA, January 2017.