aboutcode/scancode-toolkit

Project Proposal
GSoD 2019

Reference for Command-Line Options in scancode-toolkit and Reorganize the structure of AboutCode documentation at aboutcode.readthedocs.io

Ayan Sinha Mahapatra

Abstract

Presently, most of the AboutCode documentation is about getting up to speed with the tools and some tutorials on how to perform a basic task. A more detailed approach with the methods/options (basically how the basic units function, what the command-line options are, how different options affect the scan and outputs, how to use these options and examples on their use cases) would greatly improve the documentation experience from a user perspective. As of now, it's mostly a black-box to users as the documentation is focused on some basic use cases and tasks. Therefore, the using/contributing curve is sharper than what it ideally should be.

I propose to document these options and their corresponding effect in scan and outputs in detail, with examples. With that, I will add “Discussions” on how the code scanning works, organize the documentation in a structured manner, enforce standards through tests and continuous integration. Additionally, I’ll add several important Tutorials/How-To’s to further improve the documentation experience.

Goals to Achieve

1. **scancode-toolkit command-line options:** Create a complete guide of all the command-line options of scancode-toolkit, how they affect the scan and outputs, with user guidelines and explanations/examples. Add additional reference documentation for code contributors.

2. **Discussions explaining the code scanning:** Add a complete discussion on how the code scanning works, its various functions, and how it achieves smaller subtasks within the scan.

3. **ReadTheDocs Documentation:** Re-organize existing Documentation into aboutcode.readthedocs.io with proper tests and documentation standards (linting and others) for all future documentation contributions. Besides working on scancode-toolkit, extensively initiate and create/integrate basic documentation for other nexB open source projects and their integration. Basically, create the framework for an improved documentation experience.

4. **Improving/Adding to the existing Tutorials/HowTo's:** Add more tutorials and how-to guides to the existing documentation to make it more complete.

About Me

Name	:	Ayan Sinha Mahapatra
Website	:	https://ayansinha.dev
Link to Resume	:	Resume Uploaded at my Website Google Drive
Timezone	:	Asia/Kolkata (UTC+05:30)
Email	:	ayansmahapatra@gmail.com
Course	:	Electronics and Tele-Communications Engg.
University	:	Jadavpur University, Kolkata

Country	:	India
Obligations	:	None
Long-Running Project	:	No
Links	:	Github LinkedIn Twitter Blog Portfolio

Note: All my technical writing experience is documented in my resume, with appropriate links to my blog, which holds all my works and links to works in other formats. Also, compiling the blog by curating all my Technical Writings is a work in progress.

Introducing Myself

I'm Ayan Sinha Mahapatra, 3rd-year Undergrad at Jadavpur University, Kolkata. Contributing to open source software has been a dream to me because of the impact it has and what it means to us as a society. I'm highly motivated to pursue research and remain in academia after graduating, specifically studying computational intelligence and how to achieve intelligent behavior. As a Deep Learning/Neuroscience enthusiast, all of my work uses Open-Source libraries ranging from Tensorflow, Scikit-Learn, Numpy for Machine Learning and Data Science to MNE for EEG data processing.

I'm very comfortable with Python and have quite some experience in C/C++ as well. I have loved working with Sphinx, which is being used for Documentation at AboutCode, along with being experienced in similar technologies (like Jekyll, with which I've created my Blog).

I have also written code to develop meaningful projects myself and with friends in a collaborative environment that are available in my GitHub account. Alongside, as an effort to better equip my juniors in my university with trends in Machine Learning, I host sessions on Image/Data Processing. I'm also a part of the university's Code-Club (promotes Competitive Coding and Open Source Contributions) and Kaggle Club (promotes Data Science and Machine Learning), where I regularly host sessions and volunteer. I take an interest in explaining/blogging about tech and have experience in the same.

I may not be an exceptional individual, but I'm highly motivated to take on challenges and learn from them as I complete my objectives.

Why I’m applying for the Google Season of Docs

The concept of people from all backgrounds coming together as a team from all parts of the world to write impactful code, that anyone can use for their own cause, amazes me. Also, as I use open-source software extensively in my projects/research work, I know by heart the value of a project having good documentation support. It has significant effects on user productivity and how widely that project is integrated throughout developer communities in their own workflow. Inconsistencies/Gaps in documentation cause hours of frustration and lost time and wonderful documentation does just the opposite.

Now Google Season of Docs gives me the opportunity to create this positive impact on my own, as I work with an Organization like Aboutcode to improve their Documentation Experience. It’s also an opportunity for me to challenge myself, and I’m very much looking forward to learning a lot and gaining experience.

Why Aboutcode

I came across AboutCode while browsing the organization lists of GSoD. As AboutCode mainly uses Python, my language of choice, it seemed like a good fit for me. I was curious and then I set up the tools locally and experimented with Scancode-Toolkit with different codebases I use and Scancode-Workbench for visualization. I also learned that AboutCode is going to use Sphinx for their documentation and as I’ve considered sphinx before and tried it out quite some times, I was excited about contributing further. I noticed that the present documentation was in the GitHub wiki’s and the ReadTheDocs documentation was not updated. I asked Dennis whether I could update that and after getting a positive reply completed the task. While working I was exposed to all of the existing documentation and become aware of the shortcomings and what more could be done. This is when I decided that I’ll apply at AboutCode and realized that I was very excited about the work ahead. So here goes my detailed description of the work I propose to complete for GSoD 2019.

Contributions to Aboutcode

- [#17] Adds Travis-CI for sphinx-build and linkcheck

  This adds test scripts to check the build status of Sphinx Documentation along with adding checks for broken links. It also fixes some broken links and updates doc_maintenance.rst

- [#15] Adds Documentation from Wiki

  To the main documentation, this adds Deltacode, Aboutcode Docs and remaining pages of ScanCode Toolkit and ScanCode Workbench from the wiki.

- [#13] Adds GitHub wiki of Scancode-Toolkit and Scancode-Workbench

  This adds almost all the Wiki Pages of Scancode-Toolkit (Except the How-To pages) and Scancode-Workbench.

Project in Details

1. Scancode-Toolkit Command Line Options

Scancode-Toolkit has a host of Command Line options to customize how the scan is performed, the output format and several other options like post-scan plugins. These options currently don't have proper documentation to explain them and are only available through the "--help" or "-h" flag. This project aims to make a complete documentation that explains:

1. **All the Options available through Command Line**

   Goal: An exhaustive list of all possible options through the command line.

   Basic Overview:

   First, the **default scan options** are discussed, with an example of the output. A short graphic/description on how the scan is performed.

Hereafter, this default behavior acts as a reference to how the other options change the scan and the output.

The Options are:

- scan options:

[--license-diag] [--license-score INTEGER] [--license-text]

[--license-url-template TEXT] [--max-email INT] [--max-url INT]

- output formats:

[--json FILE] [--json-pp FILE] [--json-lines FILE] [--csv FILE] [--html FILE]

[--custom-output FILE] [--custom-template FILE] [--spdx-rdf FILE]

[--spdx-tv FILE] [--html-app FILE]

- output filters:

[--ignore-author <pattern>] [--ignore-copyright-holder <pattern>]

[--only-findings]

- output control:

[--full-root] [--strip-root]

- pre-scan:

[--ignore <pattern>] [--classify] [--facet <facet>=<pattern>]

- post-scan:

[--is-license-text] [--license-clarity-score] [--license-policy FILE]

[--mark-source] [--summary] [--summary-by-facet] [--summary-key-files]

[--summary-with-details]

- core:

[--timeout <secs>] [-n, --processes INT] [--quiet]

[--verbose] [--from-json] [--timing] [--max-in-memory INTEGER]

- miscellaneous:

[--reindex-licenses] [--keep-temp-files] [--test-mode]

- documentation:

[-h, --help] [--about] [--version] [--examples] [--plugins]

[--print-options]

These are to be discussed in detail and will contain the following information as mentioned in the next sections.

2. Initiate Versioning Structure

Goal: Initiate a versioning system to properly maintain cross-release options/API and documentation changes.

Problem:

Presently the documentation in the wiki and the ReadTheDocs pages are for older releases and need major restructuring.

Basic Overview:

The parts of the scancode-toolkit that have been updated/could be updated in version are

- Command Line Options
- APIs
- Documentation (To be initiated)

The command line options and the APIs are changed in versions and releases, and the documentation also has to follow, or it will create massive confusion for the users. The command line utility [--help] already is updated for any changes in options and could be used to replicate the versioning in the documentation.

3. How these Options can be used in different cases

Goal: This section will provide a basic summary of how the scan results of scancode-toolkit can be used in different causes and the Scancode-Toolkit options that provide such functionality.

Basic Overview:

This section gives different use case scenario examples and what options are recommended in those scenarios.

Note: This part requires significant help from the mentor in terms of inputs about and pointers to various use cases of Scancode-Toolkit.

4. What these Options change in the Scan and the Output

Goal: This section will provide a basic summary of how the scan results of scancode-toolkit can be used in different causes, and the Aboutcode tools that provide such functionality.

Basic Overview:

The options change the behavior of how the scan is performed.

A basic default case will be illustrated in the leading section [1. All the Options available through Command Line] and this section will compare the changes that all the options bring to this default scenario.

5. Output Formats and their examples

Goal: This section will provide a basic summary of how the scan results of scancode-toolkit can be used in different causes, and the Aboutcode tools that provide such functionality.

Basic Overview:

Scancode-Tool has flags to specify different output formats in which the scan results will be generated. These are -

- csv
- html
- html-app
- json
- json-pp
- jsonlines
- spdx-rdf
- spdx-tv

This part will

- explain in detail the output formats
- give examples on the output formats
- give other links corresponding to the output format and its use

how scan results are stored in the output files.

This also links to How these different formats are generated, which will be explained in [2. Discussions explaining Code Scanning].

6. **Business Use of Scancode Output Formats**

Goals: Explain the Business Use cases of Scancode Output formats

In the GSoD ideas list, Scancode Output Formats is mentioned as a reference idea. This section implements the same.

Note: This part requires significant help from the mentor in terms of inputs about and pointers to various business use cases of Scancode-Toolkit.

7. **How these outputs are used by other AboutCode projects for more analysis**

Goal: This section will provide a basic summary of how the scan results of scancode-toolkit can be used in different causes, and the Aboutcode tools that provide such functionality.

Basic Overview:

- Scancode-Workbench

  This part explains visualizing results with the desktop app and pointers to scancode-workbench documentation for more support on the same. Will add required documentation to scancode-workbench if necessary.

- Deltacode

  How scancode results are taken by Deltacode to determine file-level differences between two codebases.

2. Discussions explaining the Code Scanning

This is an “Explanation” type of Documentation for Aboutcode. It presents the whole process of scanning code-repositories and generating results. This part of the Project will also use flowcharts/diagrams to explain process flow diagrams and is meant to make the project more understandable to the users as a whole.

Scancode-Toolkit performs the scan on a codebase in the following steps :

1. **Collect an inventory of the code files and classify the code using file types,**
2. **Extract files from any archive using a general purpose extractor**
3. **Extract texts from binary files if needed**
4. **Use an extensible rules engine to detect open source license text and notices**
5. **Use a specialized parser to capture copyright statements**
6. **Identify packaged code and collect metadata from packages**
7. **Report the results in the formats of you choice (JSON, CSV, etc.) for integration with other tools**

There are components of the scancode-toolkit which are used in these steps to achieve specific tasks. These are - [cluecode] [commoncode] [extractcode] [formattedcode] [licensedcode] [packagedcode] [plugincode] [summarycode] [textcode] [typecode] and the main [scancode].

This section proposes to explain the following:

- What are the smaller tasks performed in a typical scan
- What components of scancode-toolkit do those smaller tasks
- How these components complete the smaller tasks
- What external libraries are used for what purposes (optional)
- Link to changing behavior because of options

Additionally, links to code APIs explained in the developer documentation can also be added to facilitate easier onboarding for developers.

3. Reorganize the structure of AboutCode Documentation

This part includes a host of changes to the Aboutcode Documentation

- **Versioning system**

  In [1. Scancode-Toolkit Command Line Options -> 2. Initiate Versioning Structure] the issue of versioning the Command Line options are mentioned. The same is necessary for other parts of the documentation also which contain version specific commands/information that would otherwise create confusion.

- **Setting Documentation Standards and Tests**

  The documentations already have tests for sphinx-build (builds all the pages and checks for Sphinx syntax errors throughout) and link check (Checks all the links to other webpages from the documentation) with Continuous Integration through Travis-CI. (Added by me in this Pull Request #17)

  Now it needs more checks for specific linting in reStructured Text and other standards. This could be achieved with restructuredtext-lint but needs more research and will be done as a part of my GSoD project.

- **Adding a “Getting Started” Section**

  This will act as a starting section for newcomers and will contain a compilation of the most basic and important documents to get started with Aboutcode Projects.

  Every Aboutcode Project will have this section including Scancode-Toolkit, Scancode-Workbench, Deltacode, and others.

- **Restructuring According to the 4 Document Functions**

  The existing Documentation isn’t explicitly structured in the 4 document functions - Tutorials, How To’s, Reference and Explanations. I propose to structure those accordingly, adding more information/explanations/pointers whatever necessary. This holds for all the AboutCode projects and their documentation. Below are two examples of the Scancode-Toolkit documentation restructuring I propose and would like to carry on in this project. Similar changes will be carried out on the rest of the documentation.

- **Restructuring the Development Page (Scancode-Toolkit)**

  More info on the Code/APIs could be added to make it more developer friendly.

  There can be links to the [2. Discussions explaining the Code Scanning] section above. This links the explanation of how the scan works to the code it uses to perform the scan.

  Like these folders contain different parts of scancode-toolkit, their individual use can be elaborated with the APIs, in conjunction with the Discussion on how scancode works.

  - [cluecode : plugins for scanning licenses, copyrights, urls, emails]
  - [commoncode : helper classes and functions]
  - [extractcode : extracts different archive formats]
  - [formattedcode : output formatting for different output file formats]
  - [licensedcode : licence detection code]
  - [packagedcode : parsing various package formats]
  - [plugincode : classes for the plugins architecture]
  - [summarycode : summarizes scan on detected licenses]
  - [textcode : handles text parsing]
  - [typecode : handles file type determinations]
  - [scancode : CLI and API to scancode, the core part]

  This subsection will contain detailed information/APIs on these parts of scancode-toolkit in subsubsections accordingly.

  The Development guidelines will be there on another page or another section having smaller subsections.

- **Restructuring the FAQ page (Scancode-Toolkit)**

  The FAQ page at present has questions that can be better answered and should be structured as separate How To's, Tutorials and Reference documents separately.

  - How does ScanCode work?

    This issue is referenced in [2. Discussions explaining the Code Scanning] and will be an entirely separate section in much more details.

  - How to Add New License Rules for Enhanced Detection?

    This issue is already discussed before in Improving the existing How-To's, documentation will be moved there.

  - How to add a new license detection rule?

    This could be made into another "How To" post separately and could be elaborated on.

  - How To get started with Development?

    There's already a separate development page and the information overlaps quite a lot. The restructuring of the development page has already been discussed above.

  - Steps to cut a new release

    This can be transformed into a seperate "How To Cut a new release".

  - Find more FAQ questions that answer generic questions about the project and don't fall in the "How To"/"Tutorial" categories.

4. Improving/Adding to the existing Tutorials/HowTo's

The main changes to the existing Documentation under this part of the project will be

- **Improving the existing How-To's**

  As an example, How to Add New License Rules for Enhanced Detection references to an issue, a related PR and another question on the FAQ page.

  All the information about adding new license rules could be bought under one page, from all other sources and adding every possible support that a user may need.

  The page How To Run A Scan is actually another Tutorial opportunity and should be improved to make it a newcomer's ideal starting point.

- **Adding new Tutorials:**

  There are opportunities like Scan a Codebase and Analyze the Results mentioned in the GSoD ideas list which will be an important addition. Having added how the scan works as an "Explanation" before this is easier to add afterward and will be more complete together.

- **Adding new How-To's:**

  The present documentation structure there isn't explicit "Tutorials" but there are pages like How To Run A Scan as mentioned above which are more tutorial opportunities and could be remodeled.

  Additionally, there are other opportunities like How To Get the License Clarity Score of a Package and How To Discover Licensing Issues in a Software Project (and how to take advantage of license policy support) mentioned in the GSoD ideas list which will be an important addition.

Project Timeline

Before the Community Bonding Period

- Setting Up the doc development environment and the aboutcode tools locally and experiment more with all the command line options and outputs
- Getting familiarized with the codebase and how the scan is performed
- Reorganize the Documentation in wiki's of specific Aboutcode Projects to host them at aboutcode.readthedocs.io
- Add Sphinx Build and Link Check Tests for CI
- Discuss and draft the project proposal with my Mentor

Community Bonding Period [August 1 - September 1, 2019]

- Fix version problems in the documentation and initiate the structure.
- Finalize the Project in a more detailed manner, make mockups of what has to be done
- Finalize the documentation structure, write more tests and linting guidelines.
- Create an Execution Plan in detail.
- Discuss my project in details with the community and integrate their feedback.

Documentation Development Begins [September 2, 2019 - November 22, 2019]

Documentation Development (September 2 - November 22)	
September 2nd - September 9th	Work on advanced Tests for Sphinx linting and setting documentation standards
September 10th - September 14th	Structuring the Command Line Options Reference and document the default case
September 15th - September 19th	Document Command Line Options and examples

September 20th - September 25th	Documenting how the different options change the Scan/Output
September 26th - October 1st	Document Different Output Formats with Examples and How these Outputs are used for visualization and analysis
October 2nd - October 5th	Work on “Discussion” part of Scancode-Toolkit (Sections 1-5)*
October 6th - October 13th	Remaining parts of the Discussion (Sections 6-7)*
October 14th - October 21st	Explain Various APIs inside scancode and support for different Plugins
October 22nd - October 25th	Integrating all this Documentation Together and Initiate Versioning
October 26th - November 6th	Additional Tutorial Documentation
November 7th - November 18th	Additional How To’s Documentation
November 18th - November 24th	Buffer Period**
Final Week: November 25 - 29	Working on submitting the final work product, report and mentor evaluations.
Final Submission	

* Here Sections 1-5 and 6-7 refer to the steps of the Scan mentioned in [2. Discussions explaining the Code Scanning]

** This Buffer Period is reserved for unplanned events or any other part taking more time

Commitments

During the project timeline, i.e. 2nd September - 29 November, I have no other prior commitments and can work for a full 42 hours per week in a regular work pattern. Will be available in all the communication channels throughout, even in the days after proposal submission and prior to officially beginning doc development.

As I'm in my last year of college, we have very little college-related work this semester (less than 10 hours per week), as placements/interviews are to be held in this period (which I will not be participating in). After college I'd be working for the same research lab I'm presently interning at, so I'd be completely free from those and all other commitments in this period. Thus I can fully focus on Google Season of Docs and quality documentation development for AboutCode.

Expectations from Mentors

During/in the time prior to Doc Development, my expectations from the mentors will be as follows:

- Helping me Understand the Codebase when required if I'm unable to understand on my own. (I'm already familiar with major structures and even some specific nitty-gritty details)
- Provide me with necessary pointers/links if I need to pick up extra concepts regarding my work when required if I'm unable to find the same on my own.
- To formulate more broadly the area of work in accordance with the existing plans of the organization, and have discussions whenever important decisions are taken (like say setting standards for the documentation).
- To be straightforward about the direction/quality of work and give feedback for path correction whenever necessary, throughout the work period.
- Take time to review my work and integrate it into the existing documentation.

Long Term Goals after GSoD

GSoD is only a stepping stone for me into Open Source Contributions. My GSoD project will improve the documentation experience and general structure of scancode-toolkit, but after that, there'll be plenty of scopes to improve the documentation. There's also scope for onboarding others to contribute to the documentation, guiding them through the structure and tests wherever necessary. I can also start contributing code to engage myself more in the open-source community, to Aboutcode or even to more organizations. I'll continue my blogging and other community endeavors at my university to mentor younger minds into open source.