Discography Web-Scraping

multiple wiki pages from a list of artist names with Python, BeautifulSoup and langdetect

by Mark Harbison mark.f.harbison@gmail.com

www.github.com/mfh92/discogSearch | www.harbison.one

I personally chose 45 music genres that interested me from more than 300 available, mostly on https://en.wikipedia.org/wiki/Lists_of_musicians . I copied and pasted the artists' names into a single list and took time to verify the spelling and disambiguation of each of their wikipedia urls. In some cases, this increased the size of the list. For example, searching for *one* band called 'Tempest' resulted in 4 different artists that all interested me:

- Tempest_(UK_band)
- Tempest_(musician)
- Tempest_(Celtic_rock_band)
- Tempest_(Christian_rock_band).

It may be possible for code to automatically test every possible suffix, but there would be *a lot* of them to attempt, without a guarantee of a complete list.

Care must be made to use the correct syntax in a url. For example, `&` becomes `%26` or `and`, depending on the artist. Some urls are displayed differently than they are in code such as when `Desorden_Público` with an international character becomes `Desorden_P%C3%BAblico` after it is copied and pasted.

The basic strategy is to append each artist name to https://en.wikipedia.org/wiki/ and let `requests` and `BeautifulSoup` create a **Soup** object with all of the html information. Then a `bs4.element.ResultSet` subset list of tags is found with `Soup.find_all_next()` (typically from the 'Discography' section to just before the 'References' section).

Actually, this program allows for a webpage to use many different content layout formats. There are 17 possible keywords similar to 'Discography' to define the start point and 18 similar to 'References' to define the end point. Start words are found with `tag.get('id')`. End words could be either based on a tag's 'id' parameter or based on `tag.name in ['p', 'dt']`, whichever occurs first.

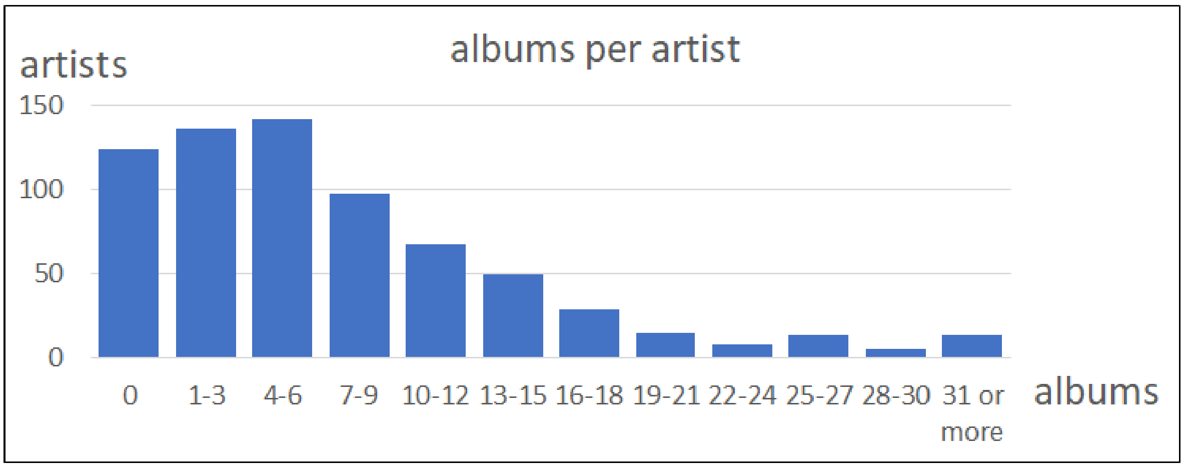
Then it finds the name and index for each section within this subset. After each section is identified as 'ok' or 'skip', then `<i>` tagged album name data are pulled from the 'ok' sections.

Some of the bad data from a section is discarded if either its tag or the parent of its tag contains an unwanted keyword. For example, since the data is based on `<i>` tags like `<i>Awesome Music</i>` which is nested in `<li><i>Awesome Music</i> EP (2017)</li>`, then `('EP' in tag.text) == False`, but `('EP' in tag.parent.text) == True`, meaning that this data will be deleted (only full-length albums are allowed, not 'Extended Play' ones).

On some webpages, this kind of deleting is not necessary since the data was neatly contained under a section heading 'EPs' (so the entire section was skipped). But since there is no enforced standard of including such labels at the section level, then it is necessary to check for keywords both at the section and at the album level.

My Results

A total of 5456 albums were saved in less than 14 minutes of runtime, based on 700 artists (sampled from my list of a few thousand). There were 140 (20%) out of 700 where where the artists' webpage failed to meet expectations, and 32 (4.6%) where my programming could be improved. This leaves a comfortable 528 (**75.4%**) where the webpage and the program worked perfectly well together. The distribution of counts is shown (from 0 minimum to 86 maximum) and averages to about 7.8 albums per artist.



Things that other people can improve

Of the 140 urls with webmaster-created bad data or formatting (out of 700),

- 117 had obs = exp = 0 (no data was returned, in agreement with the estimates, due to bad data + bad formatting by the webmaster) ;
- 16 had obs = exp > 0 (the same non-zero number of data were returned as expected, but not the correct data values, due to the webmaster) ; and
- 7 had obs = n/a since it re-directed to another url for a song or an album of an artist, or to an artist with a significantly different name than intended.

Another way to categorize these 133 urls (omitting the 7 re-directs) is to note that

- 84 had no section that was labeled with a keyword like 'Discography', which led to an accurate prediction of zero results for each ;
- 49 had inappropriate use of `<i>` tags, which could either return too much data, too little data and/or the wrong type of data.

It is unfortunate that some artists have not yet written a list of their recorded works in a Discography. Computers will not do it for them.

Italics `<i>` tags are good for distinguishing between types of recordings. The precise definition of a 'full album' (a. k. a. 'LP') may be up for debate, but it is generally understood to have more content than a 'single', '7"', '10"', or 'EP'. **Standard** formatting defines that album names should use `<i>`s, and shorter recordings should use plain text.

It can be frustrating that `<i>` tags are often misused:

- Many albums are missing `<i>`s. ({1, 2, 3, 3, 3, 5, 6, 7, 7, 8, 9, 10, 21, 25, 28, 33} albums were missed by 16 artists this way) ;
- Some EPs use `<i>`s instead of plain text (creating bad data) ;
- There are also many non-album uses of `<i>`s that accidentally get labeled as valid data, such as the names of books, movies, newspapers or magazines.

- The name of a record company should *not* be one of the appropriate uses of `<i>`, but was done so in 3 of these 700 urls ;
- The worst format was for 3 different webpages that used more than one `<i></i>` pair for one album, creating two or more partial results for the same album -- except this program treats a partial result as if it is the full album name. Future versions of this program will catch some of these errors by checking for line-breaks after each tag, but comma-separated errors of this kind will probably not get caught.

Things that I can improve

On my personal 'to-do' list are a few things that should significantly reduce the 32 programming errors (4.6% of 700):

- 18 were from scraping data from a sub-section that should have been connected to its parent section and skipped. I expect this to improve once the next version of this program distinguishes between `class = mw-headline` and smaller text like `<p>` ;
- 7 were removing good data that happened to contain bad keywords like "single", "movies" or "film". It may sound weird, but a significant number of musical artists use these words for their album titles. Hopefully, a better method will soon distinguish between 'film' as part of a title and 'film' as a description of the media format ;
- 5 were from scraping data from the same row of a table, but a different column than the album name; It should be possible to test if 2 results are on the same row of a table or not, and then delete the 'note' and keep the 'album name' ;
- 2 were language-based errors (1 Swedish, 1 Czech). The `langdetect` module does a good job detecting the language of most, but not all text. In one case, the English name was deleted while the non-English one was kept. In the other case, 'Laboratoř' and 'Laboratory' are 2 labels for the same 1 album so they shouldn't have both been saved.

As a result of these 32 programming errors,

- 25 observed more data than expected
  (ranging from 1 diff / 1 exp to 14 diff / 13 exp, avg. = 3.12 diff) ;
- 7 observed fewer data than expected
  (ranging from -1 diff / 1 exp to -2 diff / 31 exp, avg. = 1.14 diff) .

I also plan to correct for url re-direct issues (a separate list of 7 errors out of 700).

Conclusion

Even if the error rate (4.6%) is non-zero, the amount of time saved by using `bs4`, `langdetect` and `Python` is still worth it. Discovering great new music from artists that I have never before heard of has been very personally rewarding, thanks to these freely-available tools.

www.github.com/mfh92/discogSearch | www.harbison.one