Project Milestones (1)

Recommendation of similar articles from journal abstract analysis

Misty M. Giles
https://github.com/OhThatMisty

Overview

Scientists need to keep up both with their field in general and with their specific interests in order to advance their careers. I've started work on a system to help astrophysicists at Marshall Space Flight Center find articles related to a specified topic more quickly so they can discover what work has been done previously in a certain area, find potential new collaborators, or simply get back to more productive work. This system will search abstracts hosted by arXiv, a repository of scientific preprints, and use document similarity to search for related articles.

Data

ArXiv offers an XML API that is easily accessible when using the requests package to download small blocks of articles at a time. The output is an RSS-style feed that uses XML in the Atom 1.0 format, which can usually be converted to a JSON dictionary with a parser like xmltodict. This particular feed doesn't convert well because of its very nested structure. Here's a code snippet to show how three unnesting steps could only unnest *most* of the fields:

```
xmlDict = xmltodict.parse(r.text)
df_test = pd.DataFrame.from_dict(xmlDict)
df_conv = json_normalize(df_test.feed[1])
data = data.append(df_conv,sort=True)
```

Fortunately, other people have conquered this problem before. A user-created Python module called arxivpy did a tolerable job of downloading the article metadata (including abstract) and unnesting the data structure. Use of this module came with two tradeoffs that affected my use, a download issue and an unnesting issue.

When a user tries to download the metadata, arXiv can return empty "entry" fields, the field where any useful information is kept. If downloading with requests, code like the snippet above will break when it tries to either convert or append the download. It's obvious to the user that something isn't right. With arxivpy, however, the code just goes to retrieve the next block of articles without any warnings or messages. To get the most complete set that I could, I plotted the upload dates and looked for any gaps, using that information to tell arxivpy to try again. (Code for a user to download his or her own data to test my work is available; note that max_index and results per iteration are both set to 1,000 so this code will either work or fail, no gaps possible.)

The unnesting issue pertains to how the module treated a column called "journal_ref." This information should be filled in by the author after the original upload once the article has been approved for printing in a journal and should contain the journal title, issue number, and page numbers. I'd originally intended to use this information to filter for higher-quality recommendations, but it's a low-quality source that introduces additional human error - I couldn't know if the two-thirds of entries with a blank field were never published in an academic journal or if the authors never made the update. I'd already leaned toward not using the field when I discovered that arxivpy had replaced all entries with "No journal ref found." Some of the submitting authors put this information in the "comments" field, too, but many of the comments are a notation of how many pages and figures are in the copy they submitted.

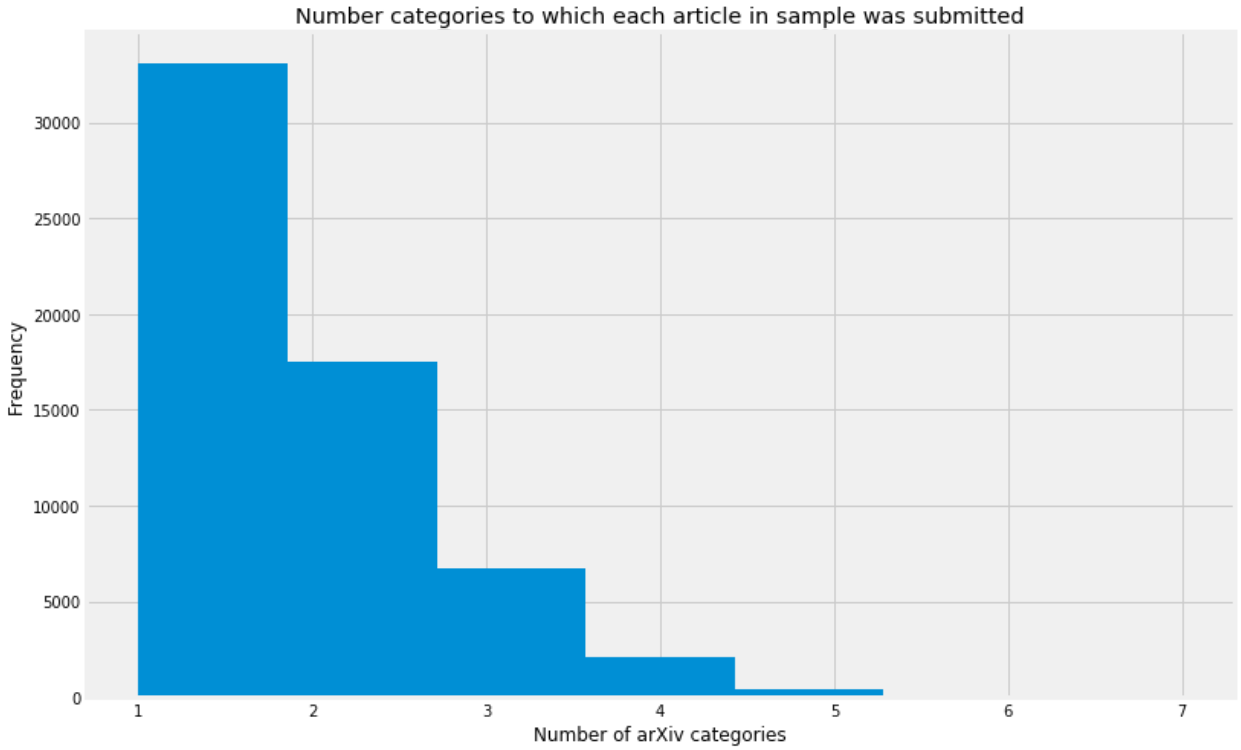
With the exception of the missing journal information, the other data fields were fairly clean and ready for analysis. The abstract itself could contain mathematical formatting leftovers from LaTeX, which will be discussed later. Two columns were removed, the id number and the pdf url, since the useful information (article identifier and arXiv's web address) was contained in the url column. The complete list of terms to which the article had been submitted (the "terms" column) were joined with a pipe by arxivpy, and I chose to use a str.contains() method to count the categories.

Data in hand, I began filtering and looking for other problems. I initially filtered to exclude submissions before 2009, duplicated abstracts, and any article that contained "has been withdrawn" in comments or summary. I also removed any article where the primary submission category was "astro-ph." The term is used in general now to talk about submissions to any of the six current categories, but "astro-ph" itself was deprecated as a submission category in 2009. These are old articles that had a more recent update date but hadn't been recategorized to the new convention.

I'd originally downloaded over 137,000 abstracts, but I found that I needed to cut the number of abstracts to analyze to 50,000 due to computer constraints. After using the same filtering as above to cut out 27 low-quality entries, I created a dataset of 49,973 abstracts that will be used in the rest of the project.

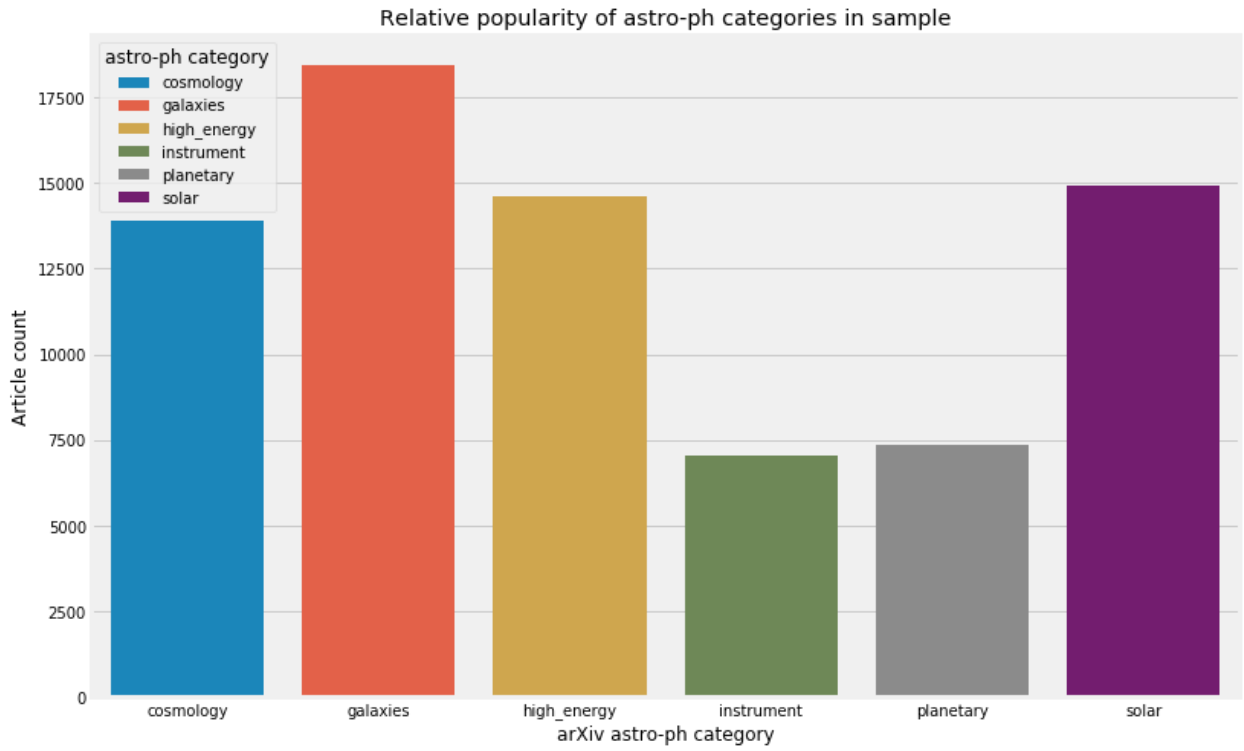
Early Visual and Statistical Analysis

The dataset of 49,973 articles spans 22 March 2016 to 17 May 2019 and has 12 columns. Two of these columns, sentences and normalized text, were created during cleaning.



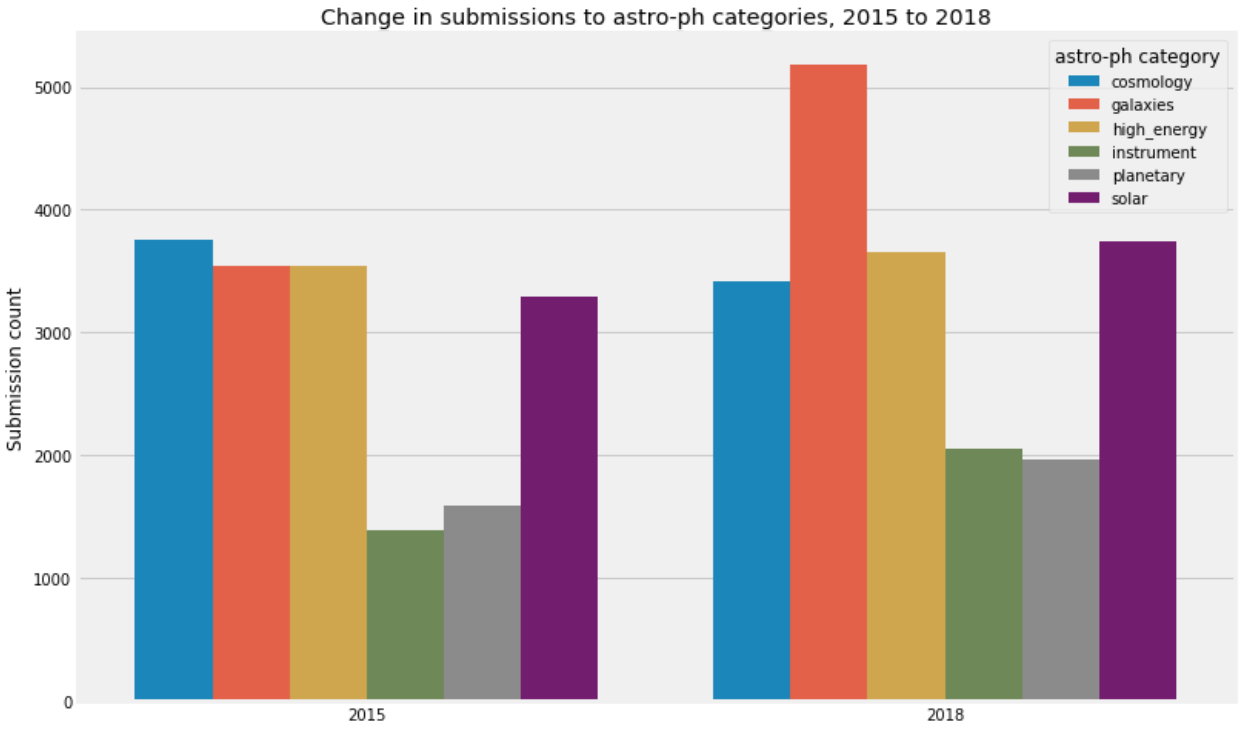
I used this initial analysis to look for some basic characteristics of the dataset. The six categories under the astro-ph (astrophysics) umbrella are cosmology, planetary, galaxies, high-energy, instruments, and solar. The articles in this dataset were submitted to at least one astro-ph category, but the primary category could be outside astrophysics. Authors submitted an article to a mean of 1.65 arXiv categories. The articles in this dataset were submitted to a min of 1 category and a max of 7 (across all fields), with a median of 1 category. One article was submitted to all six astro-ph categories.

Next, I looked at the distribution of articles across the six categories. The astrophysics of galaxies is the most popular category with 15.9k submissions. Solar/stellar astrophysics and high-energy astrophysics follow with 12.1k and 12.0k submissions, respectively.



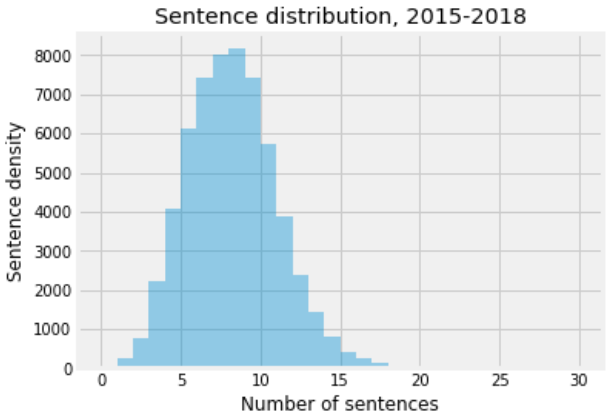
I knew before starting this project that a new mission came online in 2015 and captured the attention of the popular press. Since the first findings from LIGO were announced in early 2016, I wanted to know if there had been any noticeable difference in the distribution of articles to the categories before and after. I checked 2015 and 2018's distributions and can see that the two years do show some differences. Submissions to the astrophysics of galaxies and instrumentation categories both rose about 46%, while the overall increase in

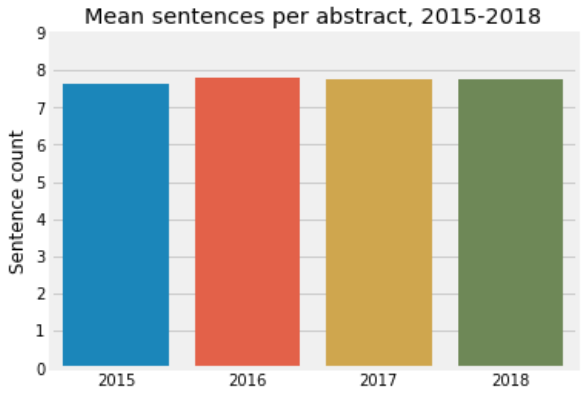
submissions was about 17%. Cosmology was the only category with lower 2018 numbers, decreasing about 9%.



Further Analysis

I'll be able to examine the structure of the text more when I begin modeling, but for now I've looked at a high-level statistic: sentence count. In the first plot on the next page, I show that the distribution of the sentences per abstract over the four years 2015 to 2018 is near normal. The distribution is skewed slightly, and that skew doesn't alter the KDE or histogram when outliers (abstracts with more than twenty sentences) are removed. The sentence counts have a mean of 7.74 and a standard deviation of 2.84. The per-year mean sentence counts of the abstracts were pretty stable from 2015 (7.62) to 2018 (7.76), as you can see in the second figure, but this difference proves to be statistically significant.





The next box contains the results of independent t-tests when the 2018 mean sentences are compared to 2015-2017. The second box then compares 2015 to 2016-2018 to help determine if 2015 is some kind of breakpoint. The results are undeniable: the 2015 sentence mean is statistically different to the other three years' means for any reasonable value of alpha, while 2018 has some similarity to 2016 and 2017.

```
Statistical difference to 2018, independent t-test
H0: year mean == 2018 mean, alpha = 0.05
2015: t = -4.1892, p = 0.0000 (reject H0)
2016: t =  0.5661, p = 0.5713 (fail to reject H0)
2017: t =  0.2729, p = 0.7849 (fail to reject H0)
```

```
Statistical difference to 2015, independent t-test
H0: year mean == 2015 mean, alpha = 0.05
2016: t = 4.6501, p = 0.0000 (reject H0)
2017: t = 4.3901, p = 0.0000 (reject H0)
2018: t = 4.1892, p = 0.0000 (reject H0)
```

The next report will cover the modeling work, where I'll determine document similarities and create some recommendations.