Final Report
Recommendation of similar articles from journal abstract analysis

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

Overview

Scientists must research others' work to advance their careers, but researchers contribute thousands of documents to their fields each year. In 2018, the astrophysics section (astro-ph) of the scientific archive arXiv received over 14,000 submissions. Although astro-ph is subdivided into six categories, the groupings are fuzzy with many subjects straddling multiple categories. Scientists at Marshall Space Flight Center have bemoaned the amount of time they must spend searching arXiv to uncover previous research in a new area.

This project uses natural language processing (NLP) techniques over a selection of arXiv abstracts to return related articles. I'll demonstrate how TFIDF and document similarity power my search.

I've provided a notebook called ArXiv_data_usertest.ipynb so readers can download sample abstracts for testing my code and also offer a pre-obtained 1,000-abstract sample. The ~50MB post-spaCy file mentioned in box 6 of my modeling notebook will be available through a link in the project README.

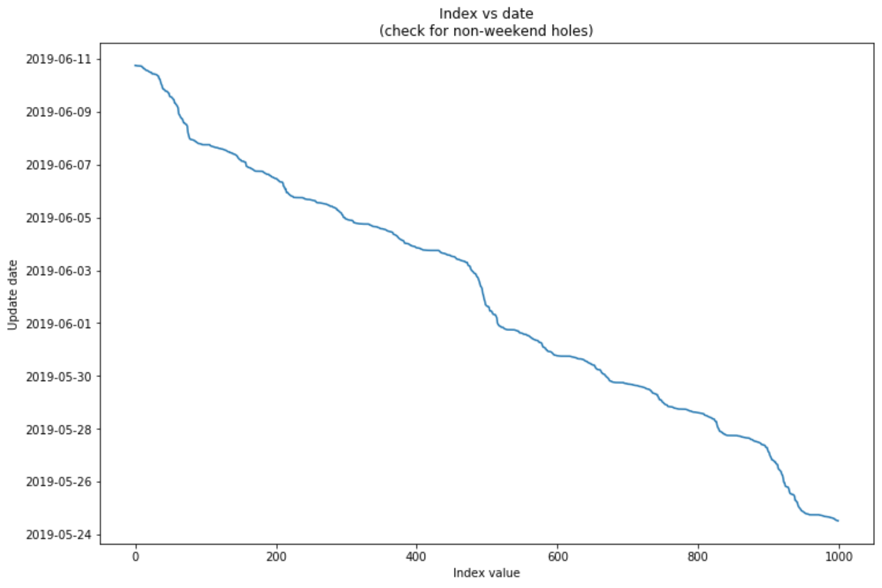
Data

ArXiv offers an XML API that is accessible through 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 JSON 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 faced this dragon before. A user-created Python module called arxivpy did a tolerable job of downloading the article metadata (including abstract) and unnesting the data structure.[1] Use of this module came with two tradeoffs that affected my use, a download issue and an unnesting issue.

ArXiv can return empty "entry" fields, the field where any useful information is kept. When I downloaded with requests, my code would break when it tried to either convert or append the empty data. It was obvious that something wasn't right. With arxivpy, however, the code moves to retrieve the next block of articles without any warnings or messages. To obtain 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 from the failed group. Here's a good download, with jaggedness from weekends but no straight lines from gaps:



[1] Although the original papers include keywords, ArXiv removes them from the metadata.

The unnesting issue pertains to how the module ignored a column called “journal_ref.” This information should be filled in by the author once the article has been accepted for publication in a journal and should contain the journal title, issue number, and page numbers. 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.

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, which will be detailed 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.

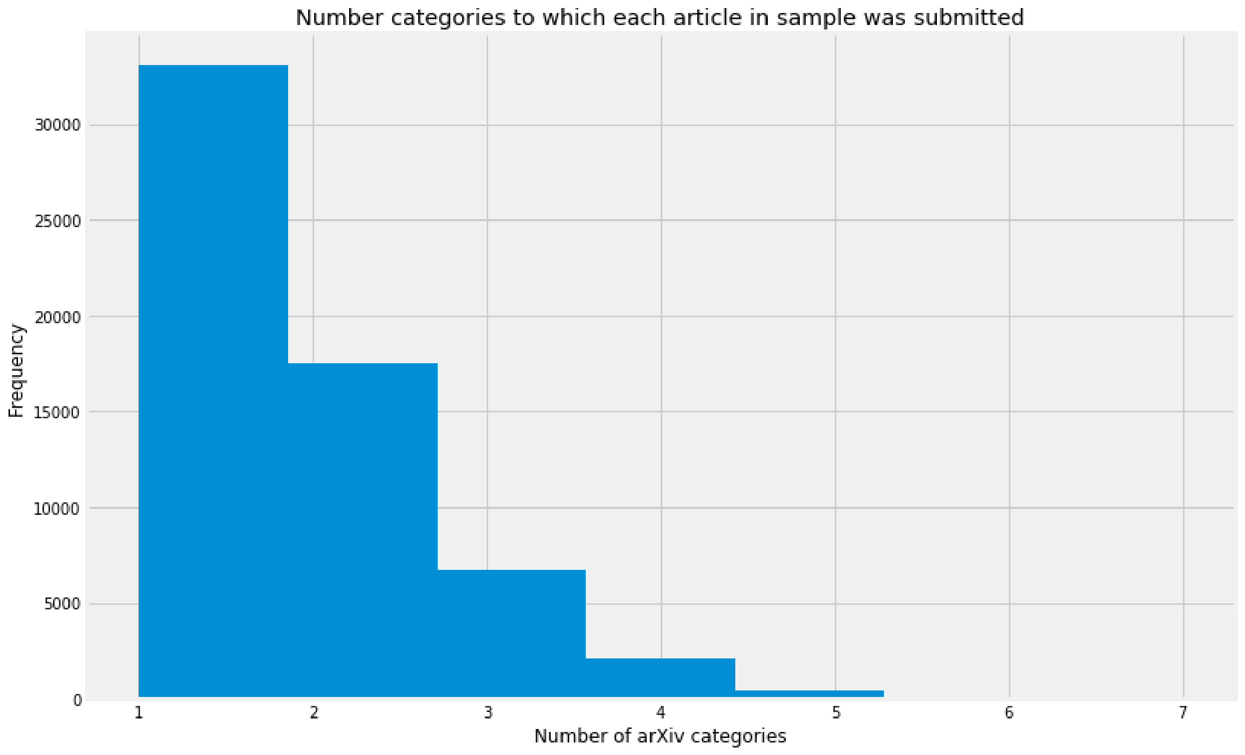
Data in hand, I began filtering and looking for other problems. I initially filtered to exclude submissions before 2009, duplicated abstracts, and any article that contained “has been withdrawn” in comments or summary. I also removed any article where the primary submission category was “astro-ph.” The term is used in general now for 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 60,000 due to computing constraints. I then used the same filtering as above to create a dataset that will be used in the rest of the project.

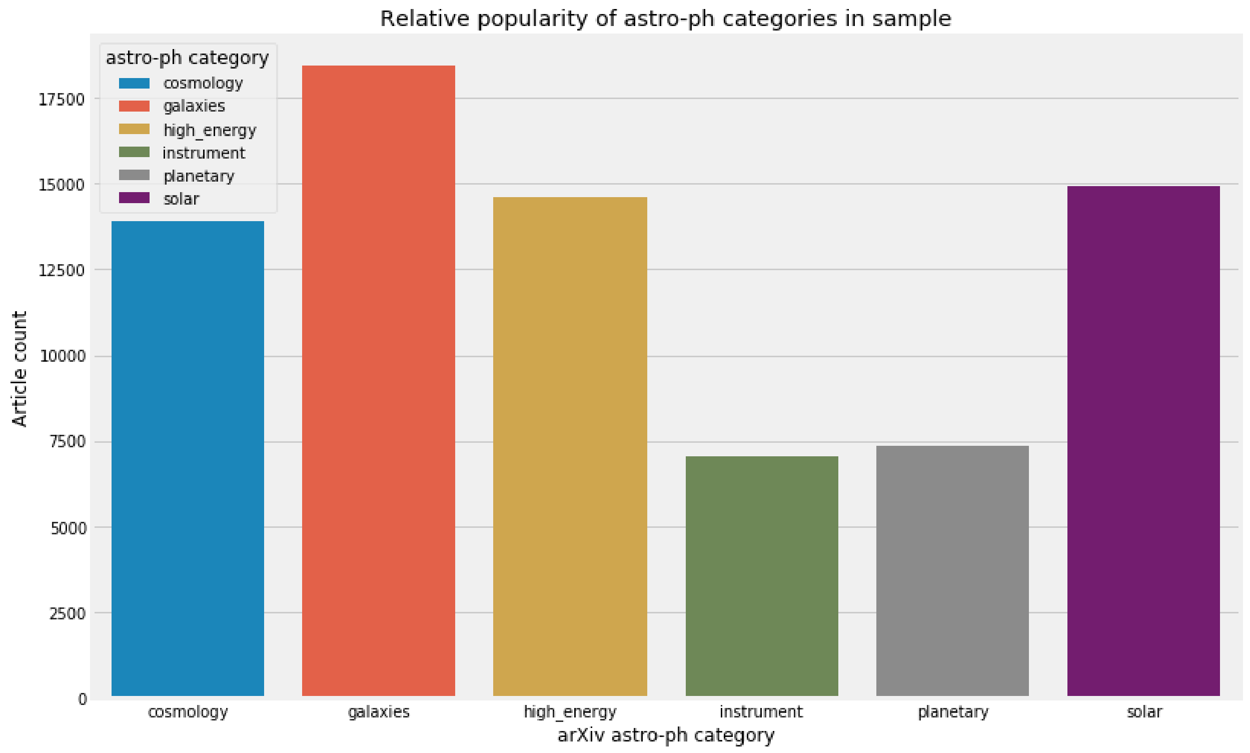
Early Visual and Statistical Analysis

The dataset of 59,972 articles has 12 columns. Two of these columns, sentences and normalized text, were created during cleaning.

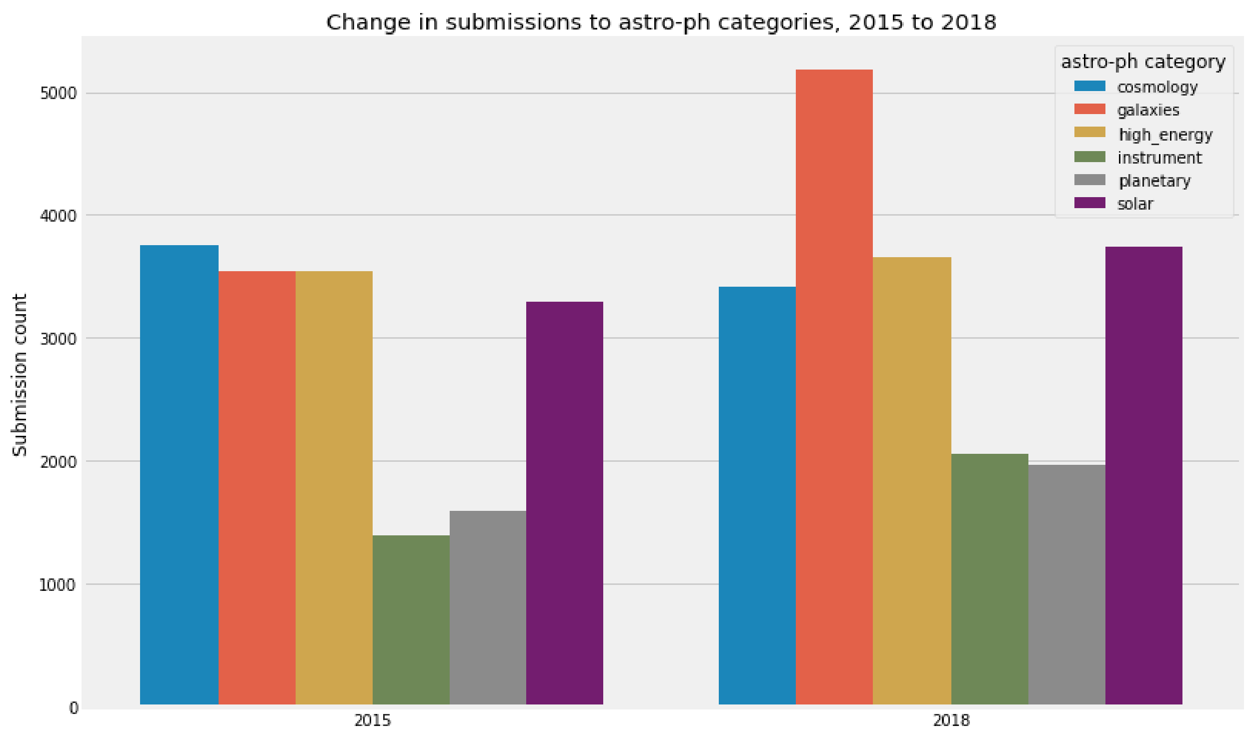
I used this initial analysis to look for some basic characteristics of the dataset. The six categories under the astro-ph umbrella are cosmology, planetary, galaxies, high-energy, instruments, and solar. The articles in this dataset were submitted to at least one astro-ph category, but the primary category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



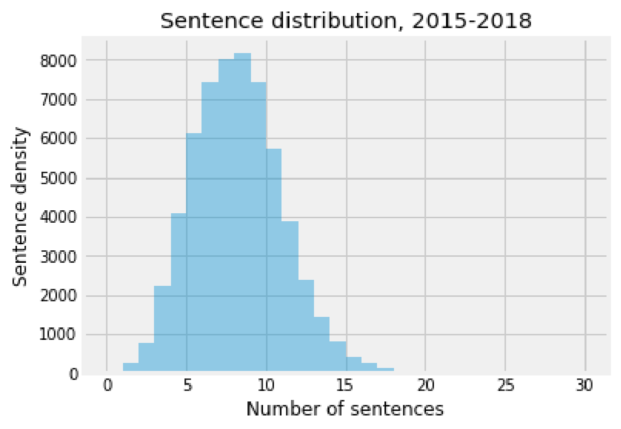
Next, I looked at the distribution of articles across the six categories. The astrophysics of galaxies is the most popular category with 15.9k submissions. Solar/stellar astrophysics and high-energy astrophysics follow with 12.1k and 12.0k submissions, respectively.

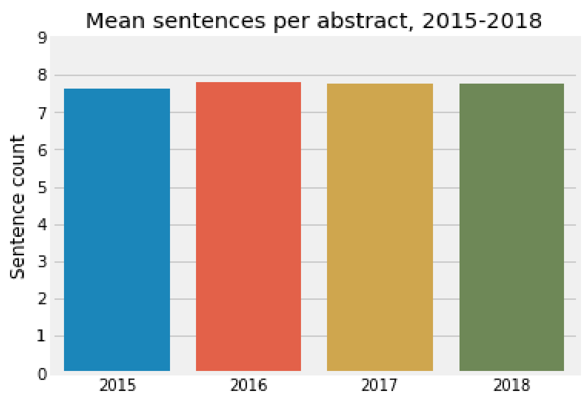


I knew before starting this project that a new mission came online in 2015 and gained attention from popular press. Since the first findings from LIGO were announced in early 2016, I wanted to know if there had been any noticeable difference in the distribution of articles to the categories before and 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%.



I also looked at a high-level statistic: sentence count. In the first plot below, 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)
```

Premodeling Text Cleaning

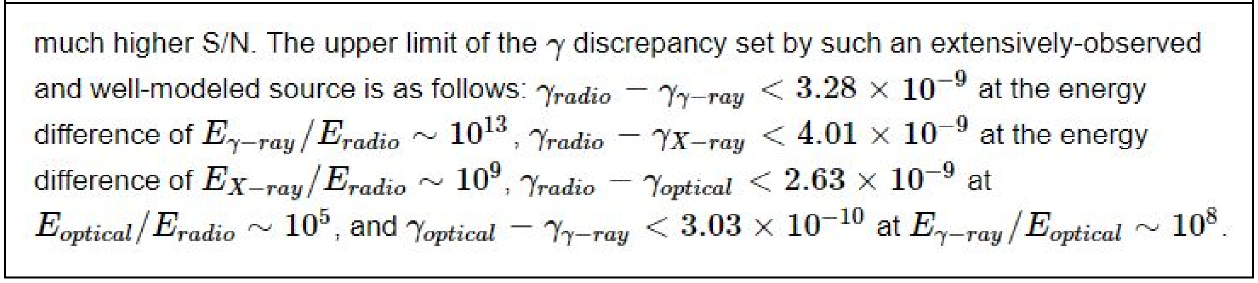
As usual in this step of the analysis, I discovered that certain aspects of the data didn't play well inside the models. Text analysis requires preprocessing, and most of the troublesome aspects of physics text could be resolved then.

In addition to the usual text preprocessing steps of lowercasing, lemmatizing, and removing punctuation, I needed to account for physics-specific lexicology. First, I used a converter to remove letters with accents (like a ç, ñ, or ü that could appear in a name) and replace as many as possible with ascii equivalents.

Scientists like equations, and they like putting those equations in their abstracts with LaTeX formatting or the matplotlib mathtext module. PDFs or HTML can process this formatting and show exactly what the author intended to write. In plain text, however, this puts symbols, numbers, and noisy words all over the place.

Here's a sample, first how the text is written and second how it appears on arXiv:

```
much higher S/N.  The upper limit of the $\\gamma$ discrepancy set by
such an extensively-observed and well-modeled source is as follows:
$\\gamma_{radio}-\\gamma_{\\gamma-ray}<3.28\\times10^{-9}$ at the energy
difference of $E_{\\gamma-ray}/E_{radio}\\sim10^{13}$,
$\\gamma_{radio}-\\gamma_{X-ray}<4.01\\times10^{-9}$ at the energy
difference of $E_{X-ray}/E_{radio}\\sim10^{9}$,
$\\gamma_{radio}-\\gamma_{optical}<2.63\\times10^{-9}$ at
$E_{optical}/E_{radio}\\sim10^{5}$, and
$\\gamma_{optical}-\\gamma_{\\gamma-ray}<3.03\\times10^{-10}$ at
$E_{\\gamma-ray}/E_{optical}\\sim10^8$.
```


much higher S/N. The upper limit of the γ discrepancy set by such an extensively-observed and well-modeled source is as follows: $\gamma_{radio} - \gamma_{\gamma-ray} < 3.28 \times 10^{-9}$ at the energy difference of $E_{\gamma-ray}/E_{radio} \sim 10^{13}$, $\gamma_{radio} - \gamma_{X-ray} < 4.01 \times 10^{-9}$ at the energy difference of $E_{X-ray}/E_{radio} \sim 10^{9}$, $\gamma_{radio} - \gamma_{optical} < 2.63 \times 10^{-9}$ at $E_{optical}/E_{radio} \sim 10^{5}$, and $\gamma_{optical} - \gamma_{\gamma-ray} < 3.03 \times 10^{-10}$ at $E_{\gamma-ray}/E_{optical} \sim 10^{8}$.

I don't know what any of those formulas mean, but I see a punctuation-removal problem. First, removing the punctuation would leave behind remnants of mathtext. Those remnants would include a lot of "gamma" - a word that can be very important in classifying high-energy astrophysics articles but is math noise here. Other Greek letters like alpha, lambda, and sigma present the same issue.

While I was unable to find a way to remove the mathtext/LaTeX as a whole from text files, I solved the problem with a series of regular expressions (regex). The text in the example could be handed by selecting all "words" that fit a pattern, '\$\S*\$', and then substituting them with a space. (The pattern stands for "find a $ and then find another $, and select everything between them as long as there aren't any spaces." This pattern would select a string like "$mathtext$" but wouldn't select "$math text$.") Some terms didn't have surrounding $s, so I utilized regex to replace patterns like "\\math." A handful of words also needed to be hard-coded into the removal code.

I still needed to remove single characters left over from the math formatting. Early efforts had the unintended effect of entirely removing "x-ray" from my corpus, an oversight that would have affected the solar, stellar, and galaxy results, for sure. (I eventually substituted "x_" for any "x-" and improved the recommendations.) After this, I could plan to remove the remaining punctuation with more custom regex.

I also used regex to test another perplexing problem. Astrophysicists use numbers to name stars and events. Some names seem so prevalent that they could come up in at least 1% of my abstracts -- Cyg X-1, GRB 130427A, GW170817. After several tests on the full set of data, I found one name that came near 1%: GW170817 at only 0.68%. At this point, I could remove the digits and any non-underscore punctuation.[2]

The rest of my text normalization was fairly standard: remove stopwords, lemmatize the tokens, and join the tokens back together as one sentence per abstract in a text file that the model could read.

Modeling

I selected TFIDF (term frequency–inverse document frequency) as a good model for the abstracts. It's easy to manipulate, and I can adjust the parameters to filter out typos and common words. TFIDF measures how frequently a word occurs in a document and compares that to how *infrequently* it appears in the rest of the documents. I used a floor of 400 abstracts to help reduce the vocabulary to something manageable on my personal computer.

The only other parameter I found reason to change was the ngram_range. Ngrams are groups of words that appear together at least as frequently as the selected floor. I tested ngram_ranges of one-to-three, one-to-four, and one-to-five words. Fewer than ten results over three words met the threshold to be included in the vocabulary. Using (1, 3) -- or unigrams, bigrams, and trigrams -- provided a vocabulary of about 2,200 words and nearly 7 million stopwords, the words and phrases that didn't meet the threshold.

TFIDF in sklearn is so inexpensive that I was able to test non-default values for other parameters until I was satisfied with the results. Most parameters worked well at default values.

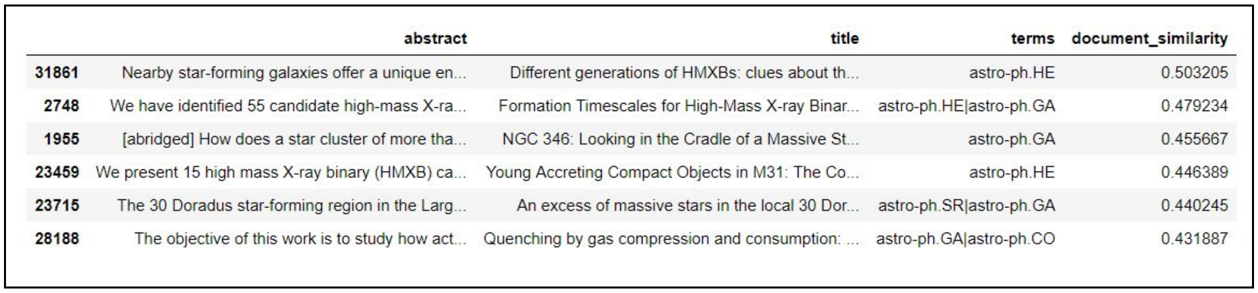
[2] Please see the "Future Directions" section for more of my thoughts on mixed-alphanumeric tokens in an astrophysics project.

Recommendation Engine

While this is an unsupervised learning project, I started with a training group (98%) and a testing group (2%). To ensure that the training data covered all dates, I shuffled the rows of the dataframe with pandas' .shuffle() ability before data normalization. I didn't attempt any stratification and make no claims about the dates in the testing group.

The TFIDF model had been fit on 58,772 abstracts, leaving 1,200 that were only transformed. These 1,200 serve as the "user input" for the recommendation engine. This engine would recommend abstracts similar to the one being read, using cosine distances calculated by the TFIDF model. It currently operates by choosing a random abstract from the test group and calculating the distances between that abstract and the abstracts in the larger group.

The engine first prints out a small table. As seen in the following screenshot, the table provides the current index (shuffle doesn't preserve the original indices), abstract, title, terms (arXiv categories to which the article was submitted), and document similarity. This is where I will use domain knowledge to check that I've been fed reasonable responses.



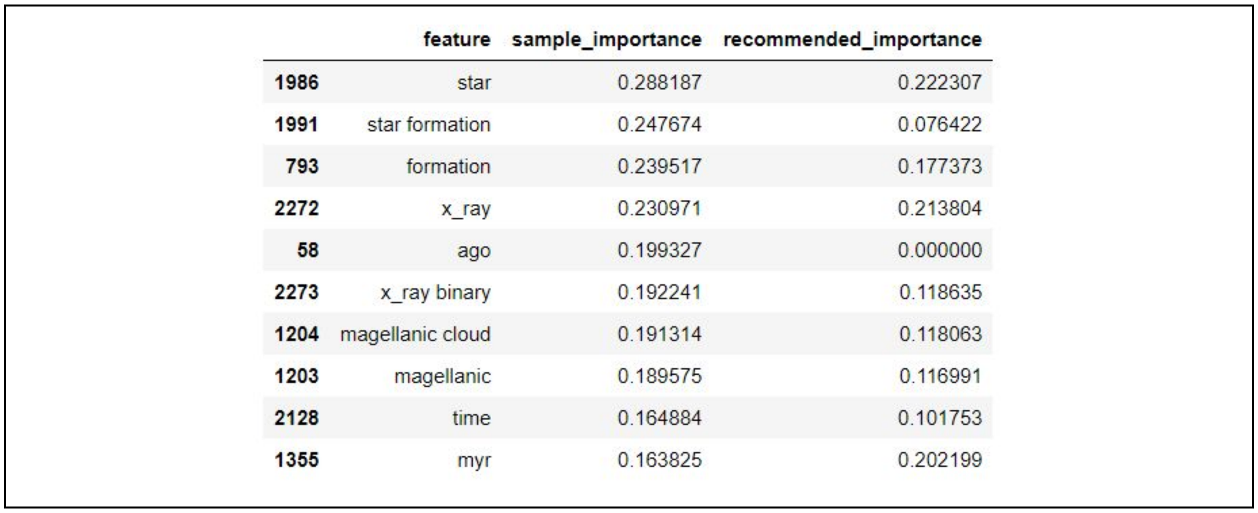
	abstract	title	terms	document_similarity
31861	Nearby star-forming galaxies offer a unique en...	Different generations of HMXBs: clues about th...	astro-ph.HE	0.503205
2748	We have identified 55 candidate high-mass X-ra...	Formation Timescales for High-Mass X-ray Binar...	astro-ph.HE|astro-ph.GA	0.479234
1955	[abridged] How does a star cluster of more tha...	NGC 346: Looking in the Cradle of a Massive St...	astro-ph.GA	0.455667
23459	We present 15 high mass X-ray binary (HMXB) ca...	Young Accreting Compact Objects in M31: The Co...	astro-ph.HE	0.446389
23715	The 30 Doradus star-forming region in the Larg...	An excess of massive stars in the local 30 Dor...	astro-ph.SR|astro-ph.GA	0.440245
28188	The objective of this work is to study how act...	Quenching by gas compression and consumption: ...	astro-ph.GA|astro-ph.CO	0.431887

I first check the document similarities. In this particular example, they're all between 0.43 and 0.50. As long as these numbers are fairly close together, I'm satisfied. The similarity numbers themselves don't matter as much -- I've seen the first result range anywhere from 0.74 to 0.38 in testing as the abstracts range from esoteric and extremely specialized to shallow and intended for general consumption -- but they do offer a sense of fit. If the similarity is too high, the engine might have provided a list of abstracts about a project, written by the project team, and created from a project template. (The Cherenkov

Telescope Array (CTA) abstracts in particular will have high similarity to another CTA abstract.)

The second check involves the categories to which the author submitted the work. If every abstract showed up with the same category represented in all rows, I'd be fairly confident that the engine had the general idea of the sample abstract and skip to scanning over the abstracts and titles. Each row in this screenshot, however, lists either high-energy astrophysics (astro-ph.HE) or physics of galaxies (astro-ph.GA). My domain knowledge is limited, so I'll check over the abstract previews and the titles. For the two marked astro-ph.HE, I see that 31861 starts with "star-forming galaxies" and 23459 mentions "M31," which sounds like a galaxy. An internet search confirms that it's another name for the Andromeda Galaxy. Now I know that all six recommendations involve galaxies.

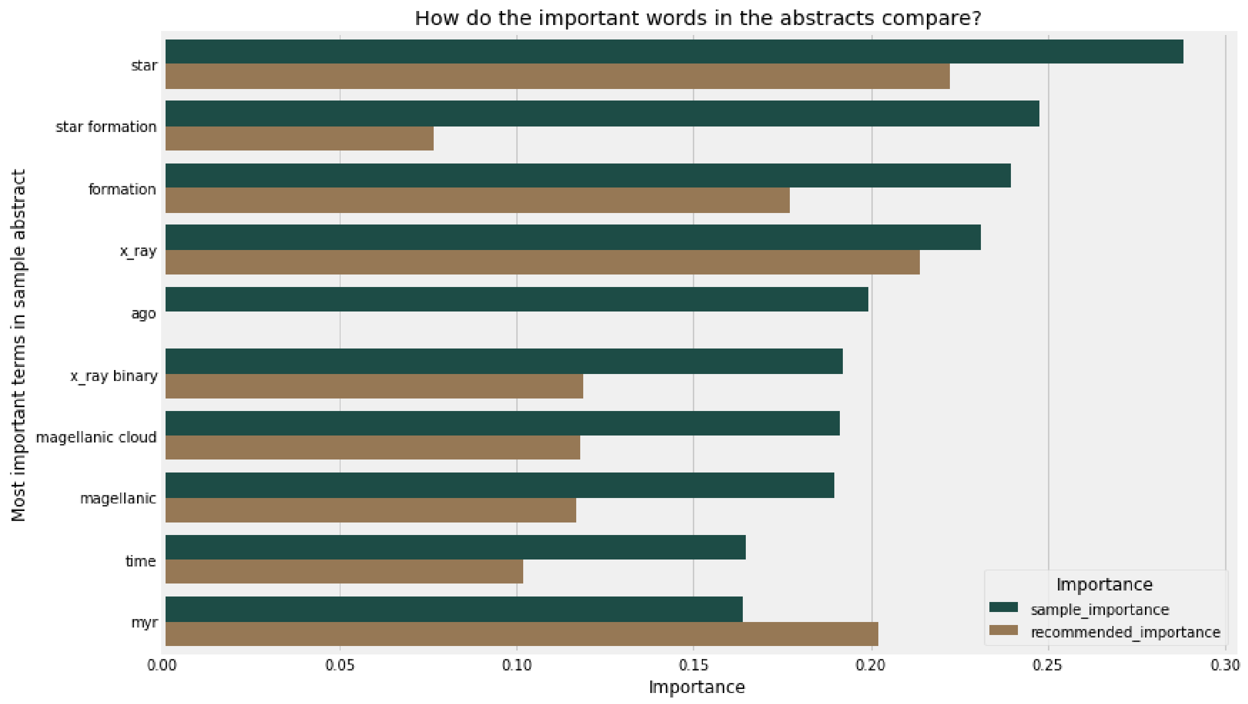
In the next screenshot, the feature importances for the sample abstract are sorted, and the top ten features are listed. Next to them are feature importances of the same terms from the recommended abstract (first result on earlier table of six) for comparison.



	feature	sample_importance	recommended_importance
1986	star	0.288187	0.222307
1991	star formation	0.247674	0.076422
793	formation	0.239517	0.177373
2272	x_ray	0.230971	0.213804
58	ago	0.199327	0.000000
2273	x_ray binary	0.192241	0.118635
1204	magellanic cloud	0.191314	0.118063
1203	magellanic	0.189575	0.116991
2128	time	0.164884	0.101753
1355	myr	0.163825	0.202199

This example shows only one term, "ago," that isn't in both abstracts, but most recommendations in testing showed fewer matching terms. This table doesn't provide much basis for checking the accuracy of the engine, but it does provide a way to compare which words were more impactful and whether the words make sense together.

Here I've plotted the relative strength of the two sets of features. The terms on the y-axis are the ten most influential sample abstract features, and the values are plotted in dark blue. The tan bars are the strength of the recommended abstract's features, for the terms they share. Reminder: This plot is unusual for showing this many matching features, and many other abstract pairs share fewer.



The five highest features of the recommendation are printed to the screen, as well. Finally, both the sample abstract and the recommended abstract are printed, along with their titles, URLs, and arXiv categories. The abstracts used in this example are available. Sample: Star-formation history and X-ray binary populations: the case of the Large Magellanic Cloud and Recommended: Different generations of HMXBs: clues about their formation efficiency from Magellanic Clouds studies.

In most production cases, articles would be added to the model when they were uploaded to a website with this type of system, and the TFIDF matrix would be updated regularly. I chose this sandboxed approach to see how the 1,200 test abstracts would fare. While writing the code, I tested with abstracts that had been in the tfidf.fit_transform() and abstracts that had only been through tfidf.transform() and didn't observe a difference in how the recommendation engine performed.

Future Directions

I want to take this work further and have several ideas.

First up, I'd like to make some refinements to the vocabulary. A few words trouble me, like "with," which currently shows up in 768 abstracts. I pushed the floor (minimum abstracts word appears in) from 600 (default value is 1%) to 400 to allow more words in the vocabulary, although the default value wouldn't fix this particular case. While writing the code I tried out different lists of stopwords, different timing of stopword removal, and filtering common words with a max parameter, but there's more to be done.

If I had more RAM/processing power, I would probably allow cases of mixed alphanumeric tokens, more important in a system where the user was able to type their own input. Removing all digits from astrophysics text when the computing system can handle a larger vocabulary would be unnecessarily restrictive. There really aren't that many common terms to play around with, but the breadth of named (or numbered) exoplanets, stars, galaxies, and events is overwhelming. Any useful search or recommendation system must include at least the scientifically significant ones in its vocabulary. This report already named some: GRB 130427 (one of the most energetic stellar deaths ever recorded), GW170817 (the first gravitational wave event to be seen in several bands of the em spectrum), and even M31 (Andromeda, one of the nearest galaxies to ours, although its significance might be my sci-fi childhood talking).

Using topic modeling to see how the interests of the community have changed or to make visualizations of the overlap between the astro-ph categories would be a fun addition to a neural-network-based improvement to recommendation system. I envision an upgrade to this engine based on word2vec or FastText, with more word-based search capabilities.

A potential future check of a topic model could involve verifying that instruments in the topic modeling match up with the category in which they do most of their work. (Fermi = HE, GOES ~= SR, Chandra = x-ray, etc.)