"Why is inside.bard.edu so poorly maintained?": An exploration primarily in webscraping and text classification on course lists

Noah Segal-Gould

A moderation project in the

department of Computer Science

at Bard College

Adviser: Sven Anderson

December 2016

Contents

CHAPTER 1

Process

1.1 Motivation

As a means of exploring computational methods, text presents a useful resource. Text data is plentiful and well-studied in attempts to perform text generation, recognition, and classification. For Signs and Symbols with Collin Jennings in Fall 2015, I had the opportunity to pursue a computational approach to understanding text for the first time. I made use of Topic Modeling [1], TF-IDF scoring [2], Markov chain generation [3], and other necessities for work in the Digital Humanities in this course. I often find myself asking relevant Digital Humanities questions, and one in particular brought me to this process: I wanted to know what constitutes an academic department. Do we understand fields of study to be best represented by their students or their professors? In search of data with which I could approach this question, I considered and accessed in parts two separate options: Bard College course lists [4] and Senior Projects [5]. If the ultimate question I wanted to pursue pertained much more to the nature of the representation of students in their fields, would I be better off pursuing completed and reviewed documents which resulted from study, or the initial documents which presented students with the option for study? I chose the latter.

1.2 Challenges

Incredibly, Bard College course lists are simultaneously inconsistent across semesters and departments in how they are represented in HTML web pages. It proved impossible to simply parse course list HTML files into groups of courses, course titles, or course descriptions. To add to the difficulty of inconsistent HTML formatting, the URLs for department course lists do not follow a consistent pattern. Departments, Interdivisional Programs, and Concentrations seem to always come and go from semester to semester or are renamed within the body of the course list itself or in the URL for that course list. The dataset I chose to use was tough to grapple with, so I chose to manually input 2254 URLs into a JSON file as input to my program.

1.2.1 Developing

For my final Introduction to Media project with Maria Cecire, I pursued a similar (albeit less intensive for lack of manual data entry) project performing text classification and topic modeling on Tweets from presidential candidates. I wanted to expand upon the ideas which resulted in that project by pursuing

webscraping using the BeautifulSoup library for Python. I'm reminded of when I was first introduced to programmatic HTML generation in Keith O'Hara's Object-Oriented Programming with Robots. Parsing and working with HTML is made especially easy using BeautifulSoup, but the course lists necessarily could not be parsed as I had hoped. Despite all my efforts, I could not extract individual courses from the course lists. The means by which that division is constituted per department per semester is too inconsistent, so I decided to use the best I had: plain text of course lists, parsed individually and undivided within departments. The Moderation data structure assumes the existence of the URLs JSON file, and through downloading of HTMLs, parsing of those HTMLs, then Multinomial Naive Bayes text classification on those compiled course lists via the Scikit-Learn library, it provides a probabilistic distribution of the classification of text from user input for departments specified by the user. The optimal parameters for the aforementioned Multinomial Naive Bayes classifier is experimentally determined using Scikit-Learn's GridSearchCV for model selection.

1.2.2 Time

I spent well over 50 hours on this project, researching text classification methods, performing data entry, documenting code, and debugging issues with the python packages I opted to use as well as ones with my dataset itself. It does what I set out for it to do, and the output serves the purpose I set out for it to serve, thankfully.

1.3 Results

When all HTMLs are downloaded and parsed, there exist a total of 2213 course lists. When only Computer Science, Mathematics, Economics, Art History, and Spanish are the left as the only remaining departments, only 203 course lists exist. If the user opts to download and parse the webpages instead of loading them from a JSON file, that process finished normally in just over 4 minutes, and should the user choose to find the optimal Multinomial Naive Bayes classifier using Scikit-Learn's GridSearchCV model selection, that process finishes in just under 70 minutes. It determined that of the potential parameters, the Scikit-Learn Pipeline which sought to contain a Count Vectorizer, Tfidf Transformer, and Multinomial Naive Bayes Classifier was best suited to the following parameters:

```
Multinomial Naive Bayes Classifier: alpha=1
Tfidf Transformer: norm='l1'
Tfidf Transformer: sublinear_tf=False
Tfidf Transformer: use_idf=True
Count Vectorizer: max_df=1.0
Count Vectorizer: max_features=None
Count Vectorizer: min_df=20
```

1.3.1 Output

I focused my tests on four words: language, book, learning, and algorithm. The program performs predictably well.

```
********************Test Starting********************


Testing on string: "language"


Number of total courselists: 2213
Number of courselists in training set: 203


OrderedDict([('Spanish', 0.25471629092886144),
             ('Computer Science', 0.24685627519140946),
             ('Economics', 0.16809236519471937),
             ('Mathematics', 0.16664622894411127),
             ('Art History', 0.16368883974089843)])


********************Test Complete********************
```

Thus, the model unsurprisingly perceives a class similarity between Computer Science and Spanish for "language." Computer Science courses often focus on programming languages, and the Spanish language taught in Spanish courses surely plays a role in the course descriptions upon which the classification is based.

```
********************Test Starting********************


Testing on string: "book"


Number of total courselists: 2213
Number of courselists in training set: 203


OrderedDict([('Economics', 0.20197044334975373),
             ('Mathematics', 0.20197044334975373),
             ('Spanish', 0.20197044334975373),
             ('Art History', 0.19704433497536944),
             ('Computer Science', 0.19704433497536944)])


********************Test Complete********************
```

The word "book" is interestingly normalized across all five departments. Contextually this makes sense because it is rare to find an undergraduate course which does not feature any book at all, and would surely prove similar in application of more departments maintained to the final training set.

```
********************Test Starting********************


Testing on string: "learning"


Number of total courselists: 2213
Number of courselists in training set: 203


OrderedDict([('Computer Science', 0.22122313583614905),
             ('Mathematics', 0.19682836882109678),
             ('Economics', 0.1948605611195235),
             ('Spanish', 0.19486056111952316),
             ('Art History', 0.19222737310370736)])


********************Test Complete********************
```

In the context of undergraduate courses "learning," like "book" is especially rare to find absent from a course list. But, reasonably, Computer Science is weighted more highly here likely as a result of the prevalence of "machine learning."

```
********************Test Starting********************


Testing on string: "algorithm"


Number of total courselists: 2213
Number of courselists in training set: 203


OrderedDict([('Computer Science', 0.24919198266933723),
             ('Mathematics', 0.1894310422213331),
             ('Economics', 0.18854707757991357),
             ('Spanish', 0.18854707757991324),
             ('Art History', 0.18428281994950282)])
```

```
********************Test Complete********************
```

Unsurprisingly, "algorithm" is especially prevalent in Computer Science and Mathematics courses. Its presence polarizes the results very clearly into those course lists which do concern themselves with algorithms and those which do not.

1.4 Conclusion

In the end, this project approaches some interesting questions pertaining to the identities of Bard College students within their fields of study. I successfully gathered the data and wrote the program to parse and use it, with the hope that it would help me understand the polarizing language that separates departments and students at Bard College. I believe it approaches that well.

1.4.1 Future Ideas

In the future, this program would benefit from more comparison between text classification methods as well as better-organized data by manually combining and renaming departments which may have changed names over time but remain the same otherwise. With more time, both should be attainable. I would really enjoy temporally classifying text, by switching around the program to classify text based on each of the now 41 semesters available through the online course list. What words most represent election years? What words most represent the interests of students and faculty at the beginnings of new decades? These questions are answerable with relatively minimal modification.

Bibliography

[1] Digital Research Infrastructure for the Arts and Humanities. Topic modeling in python. Available at https://de.dariah.eu/tatom/topic_model_python.html.

[2] scikit-learn developers. 4.2. feature extraction. Available at http://scikit-learn.org/stable/modules/feature_extraction.html.

[3] BuzzFeed. A simple, extensible markov chain generator. Available at https://github.com/jsvine/markovify.

[4] Bard College. Index of old course lists. Available at http://inside.bard.edu/academic/courses/spring2017/oldlists.html.

[5] Bard College. Disciplines. Available at http://digitalcommons.bard.edu/sunburst.html.