Building a Recommender System for Project Gutenber Ebooks Using Scikit-Learn

John Lalor

August 23, 2014

Introduction

Project Gutenberg is an online repository of ebooks. The collection includes books that are currently in the public domain, and therefore have no copyright restrictions. The collection is maintained and updated regularly, and the books are provided in a variety of formats, including text files, html files, and in some cases, pdf files. Project Gutenberg provides an online interface for users to search for and download books, but there is no way for users to identify similar books based on a selection. The large collection of free books is a perfect location for a recommendation system that will point users in the direction of potentially enjoyable literature. This project attempts to utilize these publicly available books to build a recommender system that uses content based filtering to identify similar books based on user input. Text analysis is performed using the Python scikit-learn package. Future improvements include utilizing Project Gutenberg metadata to include number of downloads as a weighted metric for the recommendations.

Working with the Gutenberg Library

Getting the Data

The Project Gutenberg collection contains over 40,000 ebooks. The books are digital versions of previously published books that are now in the public domain in the United States. This gives Project Guteneberg the ability to provide these books to the public for no cost. The books are maintained by volunteers, and new books are added on a regular basis.

For the purposes of this project, a snapshot of the Gutenberg library was copied from their servers. Because the library includes several types of files and several versions for each book, the data needed to be cleaned so that only the newest version of each book was retained for analysis. A Python script was used to identify the correct versions and build out a corpus of works [1]. The script provided approximately 35,000 of the 45,000 Gutenberg texts.

Modeling the Books

In order to make recommendations based on a book selection, the books needed to be modeled in such a way that they could be compared to each other. The books were modeled as vector space models of books over words. Books would then be recommended according to the words that are included in both works. To accomplish this, the Python scikit-learn library was used. Sckikit-learn is a machine learning software library for Python. It includes several text processing tools and functions.

Building a model of books over words involoves counting the frequency of terms in each book. Once the term counts are calculated, TF-IDF is used to compare the frequency of terms in each book against the frequency of the terms across all documents. However, it is not practical to build a model of term frequencies for 30,000-plus book-length documents in memory. Scikit-learn provides a HashingVectorizer function that will return a sparse matrix of hashed feature counts for each word [3]. This way, we can model a portion of the corpus at a time, and combine the partial matrices after they have been built in order to perform the TF-IDF analysis. With this model it is possible to create a term frequency model for all 30,000-plus documents in a reasonable amount of time. Unfortunately, the term-count model file for all books too large to process on a standard desktop, so a decision was made to limit the number of documents in the corpus to 5000 documents. This way, a TF-IDF model of books over terms could be built efficiently. The term count matrix for all documents was retained, and future improvements would include using a more powerful computer to process the entire term-count matrix with TF-IDF.

Finding Similar Books

With the TF-IDF model of books over terms for the 5,000 selected books, it is possible to compare the books and make recommendations for similar books, given a user selection. Since we have a static set of books, and want to give users fast recommendations when they access the recommender, these calculations were performed and the 20 most similar boks for each title were obtained and recorded beforehand. This way, when a user inputs a book, the recommendations are returned very quickly.

Implementing the Recommender

Putting it Online

Once the model was built and the similarities were calculated for each book, the recommender was put online, so that users could get book recommendations. A basic web app was built using the Python web framework Flask[2] (see `gutenrecs.heroku.com` for the application). When a user accesses the site, they are prompted to enter the book id of the book for which they would like to receive recommendations. During the data cleaning process, it was difficult to find a standarized method for obtaining book names and author information for each book. As the focus for this project was on the recommendations, it was decided to search via book id, which was easy to obtain from the filenames. Future updates will include using the Project Gutenberg metadata for each book to make searching for books easier for the user.

When a book is searched by the user, the similar documents are identified and returned via python code that searches the similarities file. The server code loads the similarity file and returns the twenty similar documents identified by the preprocessed cosine similarity calculations, and returns the book ids and cosine similarity values for comparison. As this is a straightforward file read and search operation, the runtime is very short, and the recommendations are returned to the user quickly. Each book id is displayed with a link to the

Project Gutenberg page for the book, so that the user can navigate to the page and begin reading.

To deploy the application to the web, Guinicorn was used as the HTTP Server. It is a straightforward server for Unix that can run applications built on a variety of web frameworks [4]. Heroku was used to deploy the application and server to allow for a simple solution to get the app on the web [5].

Testing the Model

Once the recommender was online, several books were tested, to check the list of recommendations for accuracy and relevance. Several of the books searched, and the top 5 recommendations for each, are below.

- Book id: 14127: *A Kingergarten Story Book* by Jane L. Hoxie
    1. 1599: *Cinderella; Or, The Little Glass Slipper, and Other Stories* by Anonymous
    2. 15659: *The Beacon Second Reader* by James H. Fassett
    3. 15170: *The Child's World* by Hetty Sibyl Browne, W. K. Tate, and Sarah Withers
    4. 15929: *Mother Stories* by Maud Lindsay
    5. 14241: *More English Fairy Tales* by Joseph Jacobs and John Dickson Batten
- Book id: *10380: Bible Stories and Religious Classics* by Philip P. Wells
    1. 1582: *The Bible, Douay-Rheims, New Testament*
    2. 13224: *Poems by Jean Ingelow, In Two Volumes, Volume II.* by Jean Ingelow
    3. 12759: *The World's Best Poetry, Volume 4: The Higher Life* by Gladden and Carman
    4. 14994: *Stories from the Greek Tragedians* by Alfred John Church
    5. 15874: *Old Testament Legends* by M. R. James
- Book id: 1400: *Great Expectations* by Charles Dickens
    1. 11227: *Ten Boys from Dickens* by Kate Dickinson Sweetser
    2. 12823: *Joe's Luck; Or, Always Wide Awake* by Horatio Alger
    3. 10579: *Joe Strong the Boy Fire-Eater* by Vance Barnum
    4. 12444: *Toaster's Handbook* by C. E. Fanning and H. W. Wilson
    5. 11392: *Not Pretty, but Precious; And Other Short Stories* by John William De Forest et al.
- Book id: 13055: *A General History and Collection of Voyages and Travels—Volume 09* by Robert Kerr

1. 13366: *A General History and Collection of Voyages and Travels—Volume 08* by Robert Kerr
2. 13287: *A General History and Collection of Voyages and Travels—Volume 07* by Robert Kerr
3. 13130: *A General History and Collection of Voyages and Travels—Volume 10* by Robert Kerr
4. 13225: *A General History and Collection of Voyages and Travels—Volume 06* by Robert Kerr
5. 15376: *A General History and Collection of Voyages and Travels—Volume 11* by Robert Kerr

Recommendations for these books, and others, can be reproduced at `http://gutenrecs.heroku.com`.

Observations

We can observe several things from viewing a number of recommendations of the Project Gutenberg library:

- Even though only a subset of the library (5000 books) was included in the recommender, relevant results were returned, based on the tests listed above. Some were more relevant than others, in particular, two of the top five recommendations for *Bible Stories and Religious Classics* are religious texts, while the other three are less similar on the surface. However, the stories in *Stories from the Greek Tragedians* most likely consider similar structures of the stories of the Bible.
- The recommender can identify children's books based on the words included in them. The recommendations for *A Kingergarten Story Book* are all children's text and stories of a similar nature. These books are written for children, and therefore must include words that children can understand. This limits the number of words that can be used, so the chances that these books are using similar words increases.
- The recommender can identify multiple volumes of a collected work. Each of the recommendations for *A General History and Collection of Voyages and Travels—Volume 09* is another volume of the overall work. Even though each volume most likely deals with different stories and parts of history, the overall theme of voyages and travel is enough to trigger very high similarity scores for the different volumes.

Overall the results returned from the recommender were promising. Using the HashingVectorizer to store the partial matrices made it possible to create a term count matrix for the 35,000 documents obtained from Project Gutenberg. However, only 5,000 were able to be included in the TF-IDF calculations due to memory constraints. Even with these restrictions, relevant recommendations were returned from the application. By calculation the cosine similarities beforehand, a flat file of the top 20 recommendations for each book could be read by the application for fast results to the end-user.

Improvements

While the recommender can successfully return relevant books to end users based on their selected book id, there are several improvements that can be made in future releases to improve the quality of recommendations and increase the scope of recommendations:

- Memory constraints prevented being able to calculation recommendations for all of the Gutenberg books. With more processing power, the recommender could create a TF-IDF matrix for the full term count matrix. Additionally, cosine similarity scores could be calculated across all books, so that the scope of recommendations would include the entire Gutenberg library.
- Project Gutenberg provide metadata for each of the books in their library. This data includes book and author name, published date, and number of downloads. Book and author names can be utilized in future applications to make searching for books and displaying results more user-friendly. The number of downloads can be interpreted as a measure of the book's popularity, and could be added to the model as a weight to return relevant recommendations according to how popular they are.
- Project Gutenberg adds new books to their collection on a regular basis. They update their library as new books enter the public domian, and new versions of public domain books are released. In order to keep the recommendations up-to-date, syncing with the Project Gutenberg library on a regular basis so that the model can be rebuilt with current data, and new recommendationas can be identified and provided as they become available.

Conclusion

The Project Gutenberg library is a large collection of public domain eboks. This large collection of books spans a wide variety of time periods, subjects, and literary themes. Because these books are freely available, they can be used as a large dataset for text mining and other natural language analysis. This project led to the creation of a recommendation engine based on a TF-IDF model of books over words. In the future and with enough processing power, this recommender can be expanded to include all of the books in the Gutenberg library. Using the Project Gutenberg metadata, there is also work that could be done with regard to comparing literature across time periods, comparing authors and their texts, and tracking the popularity of different types of literature through different times in the past.

Bibliography

[1] Calculating Word Statistics from the Gutenberg Corpus, Natural Language Processing Musings http://www.monlp.com/2012/04/09/calculating-word-statistics-from-the-gutenberg-corpus/

[2] Flask http://flask.pocoo.org

[3] Scikit-learn HashingVectorizer http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.HashingVectorizer.html

[4] Gunicorn Web Framework http://www.gunicorn.org

[5] Heroku http://www.heroku.com