CS 562: Fall 2019 (Class Project)

Amazon Books Recommendations Dashboard

You will be analyzing a dataset containing **book reviews** from Amazon. The data was obtained from one of the websites linked from SNAP's website and consists of two large files that contain data on book sales and reviews on Amazon circa 2015.

Your goal is to create an application - preferably a web application accessible from a browser - that will have some of the functionality described in some detail below.

Data Files

The data is available in two files that can be downloaded from the following shared Google Drive location:

`https://drive.google.com/drive/folders/1LrALq2Diz-yHTgWImA2A4XJJSrdBcbJr`

1. `review_Books_5.json`: Book reviews are in json format with **one review per line**. There are at least 5 reviews per book and least 5 reviews per reviewer.

```
{"reviewerID": "A2XQ5LZHTD4AFT", "asin": "000100039X", "reviewerName":
"Alaturka", "helpful": [7, 9], "reviewText": "A timeless classic.  It is a very
demanding and assuming title, but Gibran backs it up with some excellent style
and content.  If he had the means to publish it a century or two earlier, he
could have inspired a new religion.From the mouth of an old man about to sail
away to a far away destination, we hear the wisdom of life and all important
aspects of it.  It is a messege.  A guide book.  A Sufi sermon. Much is put in
perspective without any hint of a dogma.  There is much that hints at his birth
place, Lebanon where many of the old prophets walked the Earth and where this
book project first germinated most likely.Probably becuase it was written in
English originally, the writing flows, it is pleasant to read, and the charcoal
drawings of the author decorating the pages is a plus.  I loved the cover.",
"overall": 5.0, "summary": "A Modern Rumi", "unixReviewTime": 1033948800,
"reviewTime": "10 7, 2002"}
```

where

- `reviewerID` - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- `asin` - ID of the product, e.g. 0000013714
- `reviewerName` - name of the reviewer
- `helpful` - helpfulness rating of the review, e.g. 2/3
- `reviewText` - text of the review
- `overall` - rating of the product
- `summary` - summary of the review
- `unixReviewTime` - time of the review (unix time)
- `reviewTime` - time of the review (raw)

2. `metaBooks.json`: Information about the books with one entry per line. Note that this may contain other useful meta-information such as the price of the book and other books viewed and/or bought together.

```
{"asin": "0001048236", "categories": [["Books"]], "description": "&#34;One thing
is certain, Sherlockians, put aside your Baring-GouldAnnotated, your Folio
SocietyIllustrated-for the time being, the Oxford is the edition to curl up with
on a winter's night&#34;--The Chicago Tribune&#34;An incomparable gift book; or,
should you find it impossible to surrender up such treasures, the best of gifts
to oneself&#34;--USA Today&#34;To the true Sherlockian, this will be a treasure;
to otherwise diverted detective story fans, it is a rich lode for
discovery&#34;--Denver Post&#34;The complete and authentic adventures of the
legendary detective--expertly edited and annotated by a team of Holmes
scholars....in a handsome, boxed set....A lovely gift&#34;--The Christian
Science Monitor&#34;Here in nine volumes...are all the adventures of Holmes and
Watson. Each book has an introduction, something new and fascinating for even
the most devoted Holmesians plus a series of intelligent notes at the back of
each volume.&#34;--Oxford Times&#34;TheOxford Sherlock Holmes, a new edition of
the stories, is a splendid piece of publishing. Nine compact volumes,
beautifully produced, each with a stimulating introduction; clear type, accurate
texts, a handy chronology, a helpful bibliography. And, most valuable of all,
explanatory notes running to 50 pages or more per volume.&#34; --John Gross,
writing inSunday Telegraph--This text refers to thePaperbackedition.", "title":
"The Sherlock Holmes Audio Collection", "price": 9.26, "salesRank": {"Books":
8973864}, "imUrl": "http://ecx.images-amazon.com/images/I/51DH145C5JL.jpg",
"related": {"also_viewed": ["1442300191", "9626349786", "1602837155",
"1598879162", "1400115159", "1478396202", "1408426250", "B007PM2A4A",
"1609980603"], "buy_after_viewing": ["0312089457"]}}
```

Web Application Dashboards

Here are some suggested dashboards that your application could provide.

1. Exploratory data analysis of the books and the reviews, customized for different queries. For example, the user may wish to see the average ratings for books as a function of their sales ranks, or to search for highest ranked books whose titles contain specified search words(s), to see book

reviews by some reviewer who has at least 10 or more reviews, to observe distributions of reviews by prolific reviewers (with say, 10 or more reviews), to notice trends in reviews as a function of years/months in which reviewed, correlations between number of reviews and price etc.

2. Personalized recommendations: the application provides an initial list of a few books (at random), and invites the user to assign ratings for one or more books In response, the application recommends books to the user that are like the selected ones. The recommendations can be based on a couple of possible approaches that are described in Chapter 9 of the textbook. Remember that the reviews are very sparse in comparison with the number of reviewers and the number of books so some form of implicit or excplicit dimension reduction will be needed. Of course, you should feel free to develop other reasonable algorithms to determine books that ought to be recommended.

   - Collaborative filtering: Find similarities between the user's "ratings" and actual reviewers, and use that information to recommend books liked by those similar reviewers.
   - Content-based filtering: Use the titles, descriptions (where present), and reviews (text and rating) to create features for all the books. Use the features to find books similar to the ones selected initially as a starting point for recommendations.

Stages

Here is a suggested sequence of stages that you can go through to implement the project:

1. Set up a pipeline that creates the skeleton for the application and also an initial user inteface. At this stage, you are not actually designing the algorithms themselves for recommendations, but rather setting the stage for data input and output display. The pipeline should allow you to:

- create a user interface (start with a basic one with minimal dashboards)
- read the data from a user-specified GCP bucket
- clean up and transform the data to create appropriate internal representations
- use the interface to construct recommendations and visualize the data.

The internal representations could be PySpark dataframes, BigQuery tables, indexes for search words etc. depending on your implememtation.

2. [Optional] Create Jupyter notebooks, e.g. on `colab.research.google.com`, to test protypes of your recommendation algorithm implementation. Remember that the dat files are large and you are unlikely to get decent performance benefits in one serial DataFrame: for instance, you may have to read the data in chunks and the matrices will get too large for any

kind of processing within reasonable time. Nevertheless, you could create sample subsets and experiment on them to develop a plan for the data munging, cleaning, transformation and recommendation steps in your app. This part is entirely optional: focus instead on parts 3 and 4 below for the bulk of the project time.

3. Implement the actual data munging and transformation functions to prepare the actual internal representations of the data. Test these to ensure that you can get bare-bones results for simple dashboards like histograms of ratings etc. on your user interface.

4. Design and implement the recommendation algorithms and connect them to the pipeline so that you can see the results on your user interface.

Caution: Please make sure that you **clean up and delete cloud resources** during your testing and implementation phases. Develop a mechanism that you follow scrupulously to ensure that you do not start accumulating runaway charges for cloud resources on GCP!

Deliverables

The project will be done in groups by the following 2-3 person groups:

- Holly Wan and Shreya Bagchi
- Yi Ting Tsay, Aaayush Yadav and Rong Rong
- Ishan Karulkar, Devin Custodio and Yuxuan Wu
- Omkar Asawale, Christopher Till and Dipal Patel
- David Bushta, Swati Sharma, and Leonard Simon
- Kangan Hu, Michael McCourt, and Darshan Gada

It is suggested that you create a shared github/gitlab repository of your code so that you can add it eventually to your online portfolio of projects. Each group should ensure that design, implementation, testing and presentation efforts are divided up in an equitable manner. You will all be asked to provide a consensus statement in your project report indicating your contributions to the group effort.

Here are some guidelines:

1. Create a single GCP project for the group so that all cloud resources are maintained under that project.

2. Maintain a copy of the data in your local storage: you should only persist the data on GCP cloud storage or BigQuery tables for as short a period of time as possible (you get billed for it from your credits!)

3. You may find it useful to install the APIs for various cloud services so that you can run pipelines and scripts from your laptop.

Presentation

On **December 9th**, you will be asked to present your project in class as a group. Each group will get about 10 minutes of presentation time, 5 minutes of demo time, followed by 5 minutes of Q&A time. Every project member is **required to participate** in the presentation. I expect your project implementation to be substantially complete by Dec. 9th. I also expect that every member of a group will be able to understand and explain every aspect of the group's submission, from design to implementation and in-between.

The presentation slides should be submitted to me (one document per group) in advance of the presentation.

Report

The final report on the project will be due by **Dec. 13th** by midnight. The report (one per group) should include substantial details about your design and implementation: imagine that you are writing a blog post describing the project!

- diagrams and description of the pipeline
- description/screenshots of the user interface
- description of the ETL functions used on the data
- descriptions of the recommendation mechanism used
- step-by-step instructions on how to execute the project
- summary of the GCP resources used
- summary of the effort provided by each group member (approved and signed by the group)

The report should be about 10-15 typeset pages overall.

Resources

Here is a short list of tutorials and blog posts that you can consult (the text is clickable in the PDF version of this file). Do not blindly copy the code from the listed sites: instead, make use of the sites to develop a better understanding of how GCP-based tools can be used to create and deploy the application.

- Building a Web App Using Python Flask and Google App Engine
- Preprocessing BigQuery Data with PySpark on Cloud Dataproc
- Collaborative Filtering with PySpark
- Recommending Movies
- SVD using PySpark
- Recommendation Systems using TensorFlow
- Text Similarity with TensorFlow and Cloud DataFlow
- Recommendations using the GCP Compute Engine