The (Mini)-Book Genome Project

Miriam Boon*
Knight News Innovation Lab
Northwestern University
MiriamBoon2012@†

Christina Burghard
Northwestern University
ChristinaBurghard2013@

Noah Liebman
CollabLab
Northwestern University
NoahLiebman2016@

ABSTRACT
The (mini)Book Genome Project explores the idea of using a book's length in pages and the Library of Congress subject headings and genres as attributes in an n-dimensional space (for $n = n_{\text{subjects}} + 1$ for length) in which the Euclidean distance metric is used to calculate the k-nearest neighbors. This is designed to function as an interactive learner that makes suggestions and then incorporates feedback from the user or a user simulator, before making further suggestions.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms
Algorithms

Keywords
EECS 349, Machine Learning, Book Recommendation, Recommendation Systems

1. INTRODUCTION
Once, knowledgeable and enthusiastic staff at local bookstores served as a great source for book recommendations. But many local stores have closed, and online recommendation systems are not as flexible as a human expert.

Most of these systems learn from all of a customer's purchases or ratings, neglecting books acquired elsewhere, and including items purchased as gifts, or purchased and disliked. It also ignores the fact that users have interests in many distinct types of books, lumping all purchases into one concept space. Many also rely on collaborative filtering, which tends to recommend items that are already popular.

Pandora tackles these issues for music recommendation by allowing users to create 'stations,' and then learning what the user wants on a given station based on the attributes — the 'genome' — of station songs.

Goodreads is a social book site with over 12 million users and a mostly open API. It allows users to put books onto 'shelves' of their choosing. This project is a first iteration on a theme similar to Pandora's: recommendations based on users' Goodreads book shelves.

*Author names are in alphabetical order.

†All addresses are at u.northwestern.edu.

2. OVERVIEW
2.1 The user experience
Meet John, a Goodreads user who wants better book recommendations. He enjoys books with lots of action and monsters, but prefers his monsters to be bad guys; he's sick of recommendations for paranormal romances. He decides to try out the (mini)Book Genome Project (mBG). John logs into his Goodreads account via the mBG site. mBG fetches his shelves via the Goodreads API, and John selects a shelf he recently made on Goodreads: "Monsters Not Lovers."

Based on John's shelf, mBG finds a book it thinks might be what John is looking for, and shows him the cover and a brief description. The first book it suggests is *Twilight*, and he gives it a thumbs down. mBG incorporates that feedback into its knowledge of the shelf, and then suggests another book: *Monster Hunter International*. John hasn't read it, and he can't tell if it fits based on the information provided, so he passes. mBG notes that this book's classification is neutral, then recommends *World War Z*. John hasn't read it either, but it clearly fits the bill. He gives it a thumbs up.

2.2 Under the hood...
What's going on while John searches for more monster mayhem? mBG treats each book on the shelf as a positively labeled instance of the concept, "Monsters Not Lovers." To find books to recommend, mBG randomly selects a book from the shelf, and queries the mBG database for a list of the most similar books for which we donŠt know the class — that is, we don't know if it belongs on the shelf. (See next section for a definition of similarity.)

We could stop there and suggest A, the book most like books on the shelf. But it's unlikely to be a good choice. Instead, we use a "k-nearest neighbor" algorithm to count how many of the k books are on the shelf, how many aren't, and how many are unknown. If the majority belong on the shelf, it's more likely A does too, and so mBG recommends it. If A doesn't seem to belong on the shelf, mBG repeats the process for other similar books. And if kNN finds no book to recommend, mBG suggests the book with the highest score, for $score = pos_{\text{tot}} - neg_{\text{tot}}$.

3. DATA

Before we can create a user experience like the one described above, we need to explore the feasibility of the mBG concept. Our first step is to look at the available book data.

We chose to use Library of Congress (LOC) subject headings and genres, along with book length in pages, as attributes. Each attribute is a dimension with values ranging from zero to a weight proportional to the frequency of the attribute among all books, the inverse document frequency (IDF). The value of a given dimension, if the corresponding attribute is present for a given book, is that IDF weight divided by the number of attributes that book has.

3.1 Acquiring data

Populating our database with all the books on Goodreads would have been impractical. Instead, we used Beautiful Soup to scrape titles and authors a from user-curated Goodreads list, "Best Books Ever." Next, mBG used the Goodreads API to find the ISBNs, and sent the ISBNs to Worldcat's xISBN API to get the LCCNs, an ID used by the LOC. Finally, mBG used the LCCNs and one of LOC's APIs to get the LOC subjects, genres, and page lengths.

With each of these steps, there was attrition. For some books, Goodreads did not return an ISBN. For others, Worldcat did not return an LCCN, the LCCN was invalid, or the LOC did not have subjects. This resulted in valid data for about 54% of the books we attempted to process, and combined with the Worldcat limit on API calls per day, we were able to populate our database with only 2,165 books.

3.2 Cleaning data

The data required extensive cleaning. In some cases, XML data was not properly structured. In other cases, nearly identical terms were used, or data was entered in a non-standard style. We were able to clean much of the data, but in some cases, there was no clear way to do so programmatically, and in other cases, we lacked the expertise to make the necessary nuanced ontological judgments. During this process, we also removed attributes that only describe one book, as they would skew our results.

3.3 Calculated data

There is a power law distribution of attribute–book relationships. That is, there are only a few attributes that are commonly used across many books, and many attributes that describe only a few books. To compensate, we used a log transformation, expanding the range of IDF values.

The weight w_i (i.e., maximum value) of each attribute i is determined by an equation based on inverse document frequency: $w_i = -\log_{10}\left(1 - 0.9\frac{n_i}{N}\right)$, where n_i is the number of books with attribute i and N is the total number of books.

Then, let book A and book B be books wherein A has n attributes and length l_B, and B has m attributes and length l_B. For $X = attr(A, B)$ (the combined attributes of both books), each with weight w_i (as calculated above),

$$d^2(A,B) = (l_A - l_B)^2 + \sum_{i \cup attrs(A,B)} \begin{cases} \left(\frac{n-m}{nm}\right)^2 w_i^2 & \text{if } x_i \text{ in } A, B \\ \left(\frac{w_i}{n}\right)^2 & \text{if } x_i \text{ only in } A \\ \left(\frac{w_i}{m}\right)^2 & \text{if } x_i \text{ only in } B \end{cases}$$

This simplifies significantly when we expand it:

$$d^2(A,B) = \left(\frac{n-m}{nm}\right)^2 \sum_{attrs(A,B)} w_i^2 + \frac{1}{n^2}\sum_{attrs(A!B)} w_i^2 + \frac{1}{m^2}\sum_{attrs(B!A)} w_i^2 + (l_A - l_B)^2$$

We pre-calculated the distances for every pair of books in the database. For 2,165 books, that gave us a total of $N^2_{\text{books}}/2 = 2.3 \times 10^6$ distance calculations, which we added to an indexed table in our database. This enables us to search for books sorted by distance, which is an efficient way to fetch a book's nearest neighbors.

4. METHOD

To test our attribute space and learning algorithm, we needed to address two questions: 1. When given a book, how will we know what class it has (i.e., whether to suggest it)? 2. To test a book for class and potentially suggest it, we first need to find a book. How will we do so?

4.1 Book suggestion

mBG starts its search by randomly choosing a book, A, from the shelf. (Initially, the user-supplied books; later, includes user-approved suggestions.) Next, it fetches the l books nearest to A, where l is the total known books plus one. It removes all known books, then sends the rest to the k-nearest neighbor algorithm for classification, exiting when it finds one the algorithm classified as belonging on the shelf (i.e., positive).

The k-nearest neighbor algorithm finds the k nearest books with known classification, then counts the known positives, neutrals, and negatives. These counts are used for two metrics. First, they are used for a simple vote: whichever count is highest is the class of this book. Second, a score is calculated for the book by subtracting the number of negative neighbors from the number of positive neighbors. If none of the books is classified as positive, we note the highest scoring book and its score before repeating this process.

If the system finds a book that the kNN algorithm classifies as belonging on the shelf, it suggests the book to the user. Otherwise, this process is repeated until kNN has been run on the books nearest each book on the shelf (drawn in a random order), and then the book with the highest score is suggested to the user instead.

5. EVALUATION AND RESULTS

To evaluate mBG's suggestions, we manually classified the entire database based on the concept 'Victorian literature'. We found 49 books whose authors are broadly considered to be Victorian-era British novelists, which we coded as positive. Books that are very similar to, but not technically

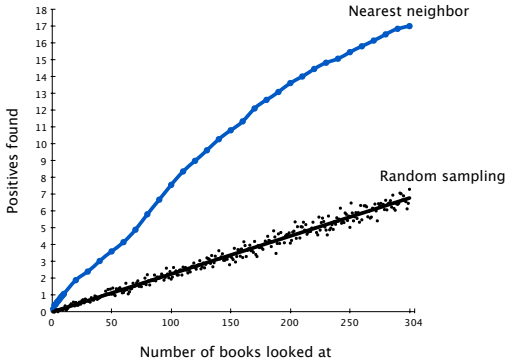

Figure 1: Concept clustering, as demonstrated by an increased likelihood of finding books on a shelf near other books on a shelf

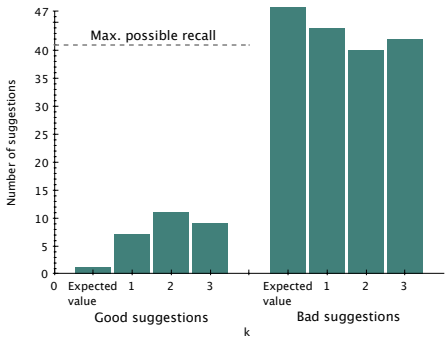

Figure 2: Summary of system performance

belonging to this concept, were coded as neutral, because we expect users to have trouble classifying them. All remaining books were coded as negative.

Armed with this fully classified dataset, the first question we set out to answer is whether distance in our feature space is a useful metric for evaluating book relatedness; that is, whether shelf concepts form clusters in our feature space. To do this, we compared the average number of positive books found in two samples of k books, for k from 1 to 300. The first sample consisted of the nearest neighbors of the shelf members, and the baseline sample was drawn randomly from the full dataset. As can be seen in Figure 1, on average, the number of positives found by randomly sampling varies with k like the expected value of a binomial distribution, with slope equal to the probability of drawing one of 49 positives from a pool of $N = 2165$ total books — $p = \frac{49}{2165}$.

The sets of books generated by fetching the k-nearest neighbors for each of the 49 books in the shelf yielded a greater percentage of other books on the shelf. This suggests that, at least for our test shelf, our features space is meaningful.

Having established that our feature space is meaningful, we created a user simulator. The user simulator, as the name implies, simulates the behavior of a user who submits a shelf to mBG, receives suggestions, gives feedback, and eventually exits the application when bored or satisfied. With a user simulator, we can perform multiple test runs and compare the results with confidence that user inconsistency is not causing variation from run to run.

We systematically tested mBG for several values of k, using our shelf and simulator. For comparison, we devised a baseline algorithm that samples books with the attribute that is most frequent among books on the shelf. But over 50 trials, this algorithm yielded no good suggestions. So instead, we compared our performance with the expected value of a random draw of size k.

As shown in Figure 2, for $k = 1$, 2, and 3, mBG makes better suggestions than a baseline of suggesting a book at random, but still achieves low recall and low precision. Out of 41 possible books to correctly suggest, mBG never suggested more than 11. Conversely, mBG made many incorrect suggestions. This is likely a result of the small size of the positives relative to the data set as a whole. We discuss strategies for compensating for this below.

6. FUTURE WORK

To expand upon the results presented here, the data should be cleaned more thoroughly — particularly the subjects. The page lengths should also be averaged over multiple editions of individual books, and books from the original "Best Books Ever" list that did not make it into this version should be included, with particular attention to books closest to the top of the list. More test sets, based on different bookshelf concepts, should also be explored.

Algorithmically, we believe that finding a way to weight the neutral and negative books so that the rare positives are not routinely outvoted would significantly improve both efficacy and speed. The team discussed several methods for accomplishing this, including scaling by shelf size, k, and/or expected value. However, none seemed clearly justifiable so we elected to leave this for future work.

Finally, it would be worthwhile to implement some version of the user interface we originally envisioned, so that we can see what it is like to interact with a system like the mBG.

7. REFERENCES

[1] W. T. Glaser, T. B. Westergren, E. F. Handman, and T. J. Conrad. How pandora generates playlists. U.S. Patent No. 11/295,339, 2005. "http://www.google.com/patents?id=jquYAAAAEBAJ".

[2] S. Maneewongvatana. A recommendation model for personalized book lists. In *International Symposium on Communications and Information Technologies*, pages 84–89, March 2010.

[3] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *"ACM conference on Digital libraries"*, pages 195–204. "ACM", November 2000. "http://doi.acm.org/10.1145/336597.336662".

[4] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *14th international conference on World Wide Web*, pages 22–32, March 2005. "http://doi.acm.org/10.1145/1060745.1060754".