Understanding Genre in a Collection of a Million Volumes

use case description by Ted Underwood
second version, May 14, 2013

To: Vijay Bhattiprolu
CC: Boris Capitanu, Mike Black (in case either of you are interested in some aspect of this)

Broad description of the problem

Large digital collections offer new avenues of exploration for literary scholars. But their potential has not yet been fully realized, because we don't have the metadata we would need to make literary arguments at scale. A subject classification like "PR - British Literature" may not reveal, for instance, whether a given volume is poetry, drama, fiction, or prose criticism. (We only have that information for about 10-20% of volumes.)

This is not a problem that can easily be solved by crowd-sourcing volume-level metadata. The dividing line between prose and verse may be fairly clear, but many interesting research questions involve subclassifications that are more contested (say, "detective fiction" or "verse tragedy"), and which may need to be redrawn in different ways by different scholars. We can't ask librarians to redo that work every time a scholar wants to test a new thesis. Moreover, the definitions of genres change over time, and there's no reason to assume that they are separated by crisp boundaries. When, exactly, does "the sensation novel" turn into "detective fiction"? It's very likely that there are a group of late-nineteenth century works with a good claim to belong to both genres.

Finally, many volumes are a mosaic of genres; volume-level metadata won't permit us to separate the "Collected Poems and Plays of Lord Byron" from the prose "Life of the Author" that precedes them in the same volume. This is particularly a problem in serials and collected works, but on a smaller scale it applies to almost every volume, if you consider that "a table of contents" and "an index" are themselves genres.

We accordingly propose to develop software that can classify volumes (and volume-parts) by genre while allowing definitions of genre to change over time, and allowing works to belong to multiple genres probabilistically. We will classify a million-volume collection (1800- 1949), make our data, metadata, and software freely available through HathiTrust Research Center, and publish substantive literary findings.

Specific work to be undertaken

I've got three broad goals.

1) One is simply to classify a large (million-volume) collection of English-language books (including nonfiction) at the document-part level. Sharing that metadata with other scholars would be worthwhile in itself. I hope to cover 1700-1949, or at least 1700-1923.

2) I also want to use this metadata to pursue my own research into English literature. I'm especially interested in the differentiation of genres from each other: for instance, can I trace the differentiation of "serious literature" from "genre fiction"? This part of the process is likely to require a lot of iterative exploration, involving both supervised and unsupervised techniques. I'll need to be able to implement a range of clustering algorithms as well as genre classification.

3) Finally, I'd like to develop an application — or a set of applications — that will allow other scholars to replicate my work, either by using different definitions of genre on the same corpus, or by applying these techniques to collections I didn't examine (for instance, to works published after 1950). I hope this will be possible because some of the data-intensive work can go on behind the scenes at HathiTrust Research Center, and users will be able to extract a condensed map of a collection using an API.

General approach

Because documents aren't homogenous, the process of mapping genre will cycle back and forth between training statistical models of classes and using those models to segment a collection. We may start with fairly broad categories: e.g., recognizing "fiction," "drama," passages of "verse," and "indexes." Once we've trained models for categories like that, we can use the models to divide volumes. With the newly segmented collection, it will become possible to develop a training corpus for subtler categories like "detective fiction" and "verse tragedy."

An important question is, How fine do we make our level of description? If we could describe works just at the volume level, we wouldn't have big problems of scale. But as I mentioned above, many works are internally heterogenous, so we probably want to think about document parts. Of course, to identify document parts, we need to locate the divisions between them. Ideally, we might do that line-by-line — and I'm open to trying it — but I suspect that won't turn out to be a viable approach. Document formats vary: I'm not confident that we can develop an algorithm to reliably recognize the line that ends one chapter and begins another. Even if we could do that, I suspect that it would be difficult to manage a map of a million-volume collection at the line-by-line level. It might especially be a problem with post-1923 texts, where we may not legally be able to extract a description of works at the level of the line. Copyright law may force us to work with a looser level of description.

Describing volumes at the page level is an appealing option, because HathiTrust data is already divided up that way — and I suspect this is the direction we're going to want to go. But there are complications. Genres aren't actually homogenous at the page level either! For instance, volumes of poetry often have prose footnotes on the same page, which should probably be considered separately. And many plays combine passages of verse and passages of prose on the same page. A genre like "the ballad opera" is in fact characterized by its *alternation* of verse and prose. So we'll want to have an ability to separate the verse parts and the prose parts of a single page; sometimes we'll consider both of those forms as belonging to a single document-part (e.g. a ballad opera). In other cases we'll separate them (e.g. poems should probably be classified separately from the prose footnotes that explain them).

I think I can make a good guess at the boundary between prose and verse at the tokenizing stage, because first lines of English poetry are capitalized up until about 1915, and that's also *the period where prose and verse tend to coexist on the same page.* After 1915, there will be an increasing number of poems without line-initial capitalization. But I'll still be able to recognize them as poems using naive Bayes to recognize poetic diction later on. I just won't attempt to do segmentation *within* a page during the tokenizing stage.

I think the boundary between verse and prose is almost the only instance where we'll need to make divisions within a page in order to characterize document parts. So right now my plan is to characterize the collection at the level of the page when I'm counting tokens, with an additional formFlag that permits us to break pages into "verse" and "prose" portions when necessary.

To make that concrete, this would mean transforming documents into a sparse table with this structure:

documentID	pageNum	wordID	formFlag	occurences
integer	short (2 bytes)	short	short	short

NOTE: The file format has been changed. When I wrote this I was very concerned to produce maximally compressed files, so I was envisioning a binary file with a fixed field length, where "documentIDs" and "words" would be converted into integer indices.

In reality, the sheer size of the data files was less important than ease of use with Hadoop, so Vijay counseled me instead to produce text files and compress them with bz2. Everything is now in text format; the documentIDs and wordIDs are stored as the original strings rather than as integer pointers.

The combination of the first four fields constitutes a unique key. formFlag will be either 0 (prose) or 1 (verse). The final field, "occurrences," tells you how many times a given word appears in that portion of that page of that document. I expect that I'll normalize words to lowercase. I won't stem words, except that I am inclined to remove final apostrophe-s.

I would also like to implement a couple of page-level features beyond wordcounts. When we're trying to identify structural features of a document, it may be useful to know:

a) how many lines are in a page (-1)

b) how many of those lines contain text rather than whitespace (-2)

and c) how many lines of text start with a capital letter (-3).

To represent these in the table, I can assign them negative "wordIDs" (listed in parentheses above) and set formFlag in those lines to -1. These features apply to the whole page, without distinction of prose or verse. They have a default value of zero, so if they're zero we don't need to include that line in the table. I've talked to Yiming Sun at HathiTrust Research Center, and I think he can generate these page-level feature counts.

Now, in my current collection of 750,000 volumes, the table I'm describing could run from 20 - 50 billion rows, and add up to perhaps 500GB. That's not huge data, but it's big enough that I suspect it will handle more easily in Hadoop. Plus, it could get bigger as we expand the collection. So think I'd like to try setting up a workflow using Hadoop, unless some reader of this document has another, easier suggestion!

Unfortunately, I don't think most humanists are going to be happy in Hadoop. So — unless someone has another suggestion — I think the implication of that is that we shouldn't expect the average humanist user to be manipulating a large collection broken down at page level. We're going to want to get the collection a bit smaller before "handing it off" to other researchers.

I think the implication is that we should do page-level analysis ourselves in order to divide volumes into document-parts. We could share that metadata with HTRC. Then, if HTRC makes available an API for feature counts, users would be able to use our metadata map to construct a database that maps a collection *at the level of generically distinct volume parts*.

In doing this, we can use the page-level metadata that exists at HTRC. But that metadata isn't consistently available for all volumes, so I doubt we can lean on it very heavily.

Instead I would suggest that we use genre classification itself to divide volumes. We'll only divide document-parts if they are separated by fairly clear and uncontroversial kinds of generic differences — genres that are very legible at the level of form. So, for instance, the *Collected Poems and Plays of Lord Byron* might have eight document-parts. Those document-parts could be mapped with a .json object that would encode information like this:

1 - pages 1-3 / *front matter* @ .8 probability
2 - pages 4-5 / *table of contents* @ .72 probability
3 - pages 6-20 / *nonfiction prose* @ .95 probability
Part 3 is *biography* @ .42 probability and *literary criticism* @ .23 prob.
5 - pages 21-48, formFlag 1 / *drama*, *verse* @ .8 probability
6 - pages 21-48, formFlag 0 / *drama*, *prose* @ .71 probability
Parts 5 and 6 together are *verse tragedy* @ .62 probability.
7 - pages 49 - 202, formFlag 1 / *lyric poetry* @ .94 probability
8 - pages 49-202, formFlag 0 / *nonfiction prose* @ .80 probability
Parts 7 and 8 are *lyric poetry with footnotes* @ .98 probability.
9 - pages 203 - 210 / *index* @ .99 probability

The classifications I've marked in gray are debatable, and they aren't kinds of classification we would use to divide the volume itself into parts. They're secondary generalizations that we could make after the volume was initially segmented. Other researchers would be free to change those judgments by running a classification algorithm on the database of document-parts. Researchers could also challenge the classifications in black, but we're going to use those classifications to divide document-parts, so if a researcher wants to challenge those boundaries, they'll need to go down to the level of the page and do the segmentation for themselves.

I'm hoping that dividing the collection into document-parts will allow us to reduce the size of the average user's database by a factor of as much as 100 or 1000. If you look at the example above, it might not look that way. 210 pages -> 9 document parts. More like a factor of 20. But (a) most volumes will have a smaller number of document-parts than this one. Most will just be front matter, ToC, prose body, index. (I don't propose to divide chapters, although that information is sometimes available in Hathi page metadata.) Also (b) in many cases scholars may be willing to ignore front matter, index, etc. They may just be interested, for instance, in the central "body" section of works -- and only interested in fiction. So doing genre classification in advance would mean that HTRC could give a user a limited number of document-parts from a limited number of volumes. A database organized like that might only run 2 or 3 GB, and I could conceivably build a Java app to manage/classify it on a user's own workstation.

So that's my current plan. I want to develop

a) A page-level classification workflow that I can use, and which might be run at HTRC to divide collections into document-parts that are distinct in (relatively) uncontroversial ways, and

b) A Java application that can interact with an HTRC API, extracting a table of feature counts at the document-part level for subtler kinds of generic classification.

If I have time, I may also build the application so that it can map a collection of documents stored on the researcher's local machine, working from the ground up. (We'd start with page-level segmentation, break the collection into document parts, etc.)

The part of this I think is most relevant for Vijay

The main things I think I'm going to need Hadoop for are

1) A fairly simple SQL-like process of selecting page-parts from the page-level table through a join with a table of volume IDs. The mapping stage might a) only select rows that match a volume ID in the table and also b) give each row a key that is equivalent to volumeID + pageNum + formFlag (I call that combination of three fields a unique "page part"). In other words, all the words on a page part will now be grouped together by the reduce operation. In many cases, page part just == page, because it's all prose (formFlag == 0) or all verse (formFlag == 1). The structural features of a page (with formFlag == -1) should be included as features in *every* pagePart.

2) Once we get these groups of features (representing page parts), we'll need to be able to run each group through a classifier We may be able to do that *in* the first reducing stage or we may need another cycle of map-reduce; I'm not sure. Right now, I'm finding that naive Bayes works adequately for classification, but I may try other algorithms. The Mahout library is possibly useful here: it's got a lot of machine-learning algorithms already implemented on Hadoop. We may need to make some adjustments to it, though.

3) The output of this reducing stage would be a probability representing the likelihood that each page part belongs to a given class.

We'll generate these lists of probabilities for a lot of different genre classes. Once we've generated all these probabilities, indicating how likely each pagepart is to belong to a range of different genres, there will be a second stage of the game where we go through each document sequentially, in page order, and decide how to segment the volume into document-parts. I'm not sure this will actually need to be done on Hadoop. Since we're no longer dealing with individual words, the scale of the data will have been reduced by a factor of something like 200. That might be small enough to just go through it sequentially ... not sure yet.

4) This is coming late in the list, but it might actually be something to tackle first. I *think* I know what the big, obvious genre categories are, but ... I might be wrong. So it would be very useful to do some page-level clustering on the dataset before I start training classifiers.

There are a range of clustering algorithms built into Mahout, and I'm going to see if I can get you a Mahout book. It seems to me that they implement k-means together with a canopy clustering feature that can help decide the appropriate k. That might be worth trying. They also implement model-based Dirichlet clustering, which I bet is useful. Let me know if other algorithms seem appealing.

I think it might be worthwhile to cluster whole pages rather than page parts. In other words, we would include all the features on a page, treating occurences of "the" in the poetry part of the page as distinct features from occurrences of "the" in the prose part. We might try tf-idf normalization on words. I would suggest converting the structural features of the page into two ratios that count as features for the purpose of clustering.

• number of text lines / number of total lines
• and number of capitalized lines / number of text lines

I would also suggest including a third ratio as a feature:

• page number / total number of pages in the volume

Since that's actually very revealing when it comes to table of contents and indexes.

This isn't actually a fully developed workflow, but I hope it's enough to give you a sense of where to start thinking and reading.