Published in: Bański, Piotr/Kupietz, Marc/Lüngen, Harald/Rayson, Paul/Biber, Hanno/Breiteneder, Evelyn/Clematide, Simon/Mariani, John/stevenson, Mark/Sick, Theresa (eds.): Proceedings of the Workshop on Challenges in the Management of Large Corpora and Big Data and Natural Language Processing (CMLC-5+BigNLP) 2017 including the papers from the Web-as-Corpus (WAC-XI) guest section. Birmingham, 24 July 2017. - Mannheim: Institut für Deutsche Sprache, 2017. Pp. 25-29

Organizing corpora at the Stanford Literary Lab

Balancing simplicity and flexibility in metadata management

David McClure, Mark Algee-Hewitt, Steele Douris, Erik Fredner, Hannah Walser
Stanford Literary Lab, Department of English, Stanford University
{dclure, malgeehe, sdouris, fredner, walser}@stanford.edu

Abstract

This article describes a series of ongoing efforts at the Stanford Literary Lab to manage a large collection of literary corpora (~40 billion words). This work is marked by a tension between two competing requirements – the corpora need to be merged together into higher-order collections that can be analyzed as units; but, at the same time, it's also necessary to preserve granular access to the original metadata and relational organization of each individual corpus. We describe a set of data management practices that try to accommodate both of these requirements – Apache Spark is used to index data as Parquet tables on an HPC cluster at Stanford. Crucially, the approach distinguishes between what we call "canonical" and "combined" corpora, a variation on the well-established notion of a "virtual corpus" (Kupietz et al., 2014; Jakubíek et al., 2014; van Uytvanck, 2010).

1 Introduction

The Literary Lab[1] is a research group in the English department at Stanford University that applies computational methods to the study of literature. The raw data behind the lab's research output is a collection of about 20 full-text corpora that contain many hundreds of thousands of novels, plays, poems, essays, pamphlets, letters, and newspaper articles spanning roughly from 1500 to 2000. These corpora come in all shapes and sizes – everything from small, ad-hoc collections of plain-text files assembled by hand for individual projects up to very large, professionally-curated collections purchased by the Stanford Library. [2] In total, these data sets comprise about 4 terabytes of raw data, and contain about 40 billion words of text.

This paper will describe a set of ongoing efforts at the Literary Lab to build a system for accessioning, organizing, and analyzing these corpora – the pipeline that runs from the raw data sets that come through the door to the final statistics, plots, and insights that appear in articles and pamphlets[3] published by the lab. This has proven to be a difficult and interesting problem because it involves navigating a set of overlapping (and at times conflicting) requirements.

2 Simplicity versus flexibility

The crux of this, in many ways, is a tension between competing desires for both simplicity and flexibility. On the one hand, we want to put everything in a single place – we want some kind of a unified data model that provides a simple, structured way to interact with the data, something that lends itself to the type of quick experimentation and hypothesis-testing that's needed in a research context. We don't want to rewrite the same ETL ("extract, transform, load") bindings onto the corpora over and over again for each project, and don't want to duplicate common preprocessing steps like tokenization, part-of-speech tagging, lemmatization, dependency parsing, etc. And, maybe most important, we often want a frictionless way to easily work *across* corpora. For example, in a study of structural changes in the American novel over the 19th and 20th centuries,

[1]http://litlab.stanford.edu/

[2]Among others – the Eighteenth and Nineteenth Century Collections Online and American Fiction corpora from Gale, the British Periodicals Online and Chadwyck Healey corpora from ProQuest, the Early English Books Online corpus from the Text Creation Partnership, and the Chicago Novel Corpus. As Tiepmar (2016) notes, there is very little standardization across text corpora used in the digital humanities, and these are no exception.

[3]https://litlab.stanford.edu/pamphlets/

we need a way to combine the ~18k novels in the Gale American Fiction corpus (1820-1940)[4] with the partially-overlapping ~10k novels in the Chicago Novel Corpus (1880-2000). We need a way to merge corpora, to bridge across them, to fuse them together into seamless collections of texts that can be easily analyzed as a unit.

But, at odds with this impulse to flatten everything out into a common data format, we also don't want to put any kind of inherent constraints on the types of questions that we could theoretically ask of the data. For example, we almost always want to preserve the domain-specific (and often very idiosyncratic) metadata that comes with the individual corpora. To give one small example – with the British Periodicals Online[5] corpus, a collection of 5 million articles from 1720-1940, recent projects have needed to make extensive use of the `<ObjectType>` element in the original XML, which classifies each article according to a custom vocabulary – "Fiction," "Review," "Advertisement," "Correspondence," "Poem," etc. At first we tried to pick a single, flexible metadata standard that could accommodate texts from all of the corpora and map these types of fields into this common schema. But this became unwieldy – taken together, the corpora have extremely heterogeneous metadata, and while it was possible to map everything into a single schema, we quickly ended up with a kind of Frankenstein format in which individual metadata fields become confusingly overdetermined and start to mean very different things in different contexts.

And, in some cases, corpora come with data that is almost impossible to fit into any kind of standardized schema designed for books or articles. For example, a graduate student in the Lab is working with a corpus of ~16k books scraped from a "fan fiction" website,[6] which includes metadata like the number of "favorites" and "follows" on the book, lists of characters, information about when individual chapters were published or updated, and even additional entity types like reviews and comments, all of which would be difficult to shoehorn into a one-size-fits-all schema. And yet, for that particular corpus, all of this information is extremely useful from a research standpoint.

[4]http://www.gale.com/c/american-fiction-1774-1920

[5]http://www.proquest.com/products-services/british_periodicals.html

[6]https://www.fanfiction.net/

3 Canonical and combined corpora

On the one hand, then, a desire to have everything in the same place; but, on the other hand, a practical need to retain a pristine copy of the original metadata for each corpus. Over the course of the last few months, we have been experimenting with a new workflow that tries to accommodate both of these requirements at once. The project is a Scala codebase that uses Spark[7] to write data as Parquet[8] tables on Stanford's Sherlock cluster[9], a 120-node HPC cluster administered by the Stanford Research Computing Center.

The key idea is to distinguish between what we call "canonical" and "combined" corpora. (This is similar to the notion of a "virtual" corpus described by Kupietz (2014), Jakubíek (2014), and van Uytvanck (2010).) Bindings for the "canonical" corpora wrap the raw, upstream data that comes from the vendor and extract exact copies of all included metadata, generally preserving the original nomenclature exactly to avoid any ambiguity about where a field came from – for example, a column in a CSV of authors called "Secondary Occupation" would become a `secondaryOccupation` field. Entities from the corpora – novels, poems, plays, authors, profiles, reviews – are represented as separate Scala case classes, which, combined with Spark's `Dataset` API, provide a typesafe way to represent the different schemas, which makes it easier to avoid errors down the line when fusing them together into unified collections. These "canonical" corpus readers also index the full-text content in a standardized way – in addition to storing the unmodified plain text for open-ended analysis, a stream of parsed tokens is also stored as an array in each Parquet row, with each token annotated with basic metadata – the original word form in the text, a part-of-speech tag assigned by OpenNLP[10], start and end character positions, and the "offset," a 0-1 value that represents the word's ratio position inside the text. (This corresponds to "Level 1" annotation under the rubric of the Corpus Query Lingua Franca, as described by Evert et al. (2015).)

These bindings onto the raw corpora produce full-fidelity, stable versions of each corpus for use in Lab research. For projects that are just using

[7]https://spark.apache.org/

[8]https://parquet.apache.org/

[9]http://sherlock.stanford.edu/

[10]https://opennlp.apache.org/

one corpus, feature extraction jobs can be run directly against these canonical Parquet tables. No constraints are placed on the structure of these initial copies of the corpora – they can be exactly as simple or complex as needed to represent the raw transmission data.

Meanwhile, for projects that need to mix and match different corpora together – for example, a project that needs a unified novel corpus from Gale American Fiction, the British Library corpus, and the Chicago corpus – a new adapter is created that generates what we call a "combined" corpus, a temporary data set that is tailored around the needs of that specific project. The code to produce a combined corpus looks very similar to the code that wraps one of the original corpora, except that the combined corpus will simply read from the Parquet tables produced for each of the canonical corpora instead of directly parsing the raw transmission data. The combined corpus provides a custom metadata schema that merges together just the specific fields that are needed in the context of the project at hand, and the extraction job writes out a single Parquet table that serves as the "working" data set for that project. Last, in addition to defining this set of mappings by which the corpora are fused together for a project, the code that generates the combined corpus is also responsible for the key step of de-duplicating the texts that get mapped into the unified schema, using the MinHash / LSH approach described by Leskovec, Rajaraman, and Ullman in *Mining Massive Datasets* (2014). In the future, we plan to wrap this up as a structured API that can easily be reused when defining a new combined corpus.

Once these Parquet tables have been saved to disk – at which point there is no meaningful distinction between a canonical and combined corpus – these datasets can serve as the basis for the actual "analysis" or "query" jobs that ask specific research questions of the data. These jobs vary widely in size and scope. More often than not, they have more in common with what might be thought of as "feature extractors" than with "queries," in the strict sense of the idea – usually, instead of directly producing a result that can be interpreted by a researcher (eg, KWIC results), these analysis jobs will generate some kind of intermediate dataset tailored around a particular set of questions, generally small enough to fit in RAM on a regular computer – often CSV, JSON, or SQLite files, or binary models that have been trained on the corpora. These intermediate datasets are then usually moved off the HPC cluster and taken up in statistical environments like Jupyter notebooks or RStudio, where the final querying, analysis, and data visualization takes place.[11]

4 Problems and future directions

One thing we have struggled with is whether it makes sense to save the "combined" corpora to disk, or if they should be materialized on-the-fly when analysis jobs are run.[12] The downside to saving them, of course, is that we end up storing duplicate copies of the texts that get included in the combined corpora. For example, if three active projects use Gale American Fiction corpus, then we store it four times – once in the "canonical" table, and three times for each "combined" corpus. Jakubíek (2014) sees this as unworkable, and in some contexts it certainly would be – for example, in a public-facing project with hundreds or thousands of users, where duplicating portions of the corpora for each user would vastly increase the storage requirements.

But, in the context of an individual research group, this hasn't been a problem. The Lab has a 30-terabyte quota on the HPC cluster (of which we've never used more than 4-5), and the storage requirements for the combined corpora are more modest than they might seem. Because the combined corpora are inexpensive to generate – the computationally intensive work is done up-front by the canonical adapters – they can be treated as ephemeral data and deleted as soon as a project ends, making it unnecessary to store more than a handful at once.

Furthermore, from a standpoint of what might be thought of as "engineering ergonomics," there are some interesting advantages to saving the combined corpora as complete, self-contained pack-

[11]To pick up on Evert et al.'s taxonomy of "approaches" to querying corpora – most of the analysis jobs run by the Literary Lab fall into approach 3, where "requirements can only be satisfied by a Turing-complete query language." (Evert et al., 2015) Which, in this context, is just an open-ended Spark job written in Scala, Python, or R, operating on the raw or annotated text content.

[12]This could be accomplished fairly easily – a "join" table could be generated that would just store foreign-key references back to the texts in the canonical corpora along with the results of the deduplication process, and the texts could then be mapped together into a unified `Dataset` at runtime in Scala.

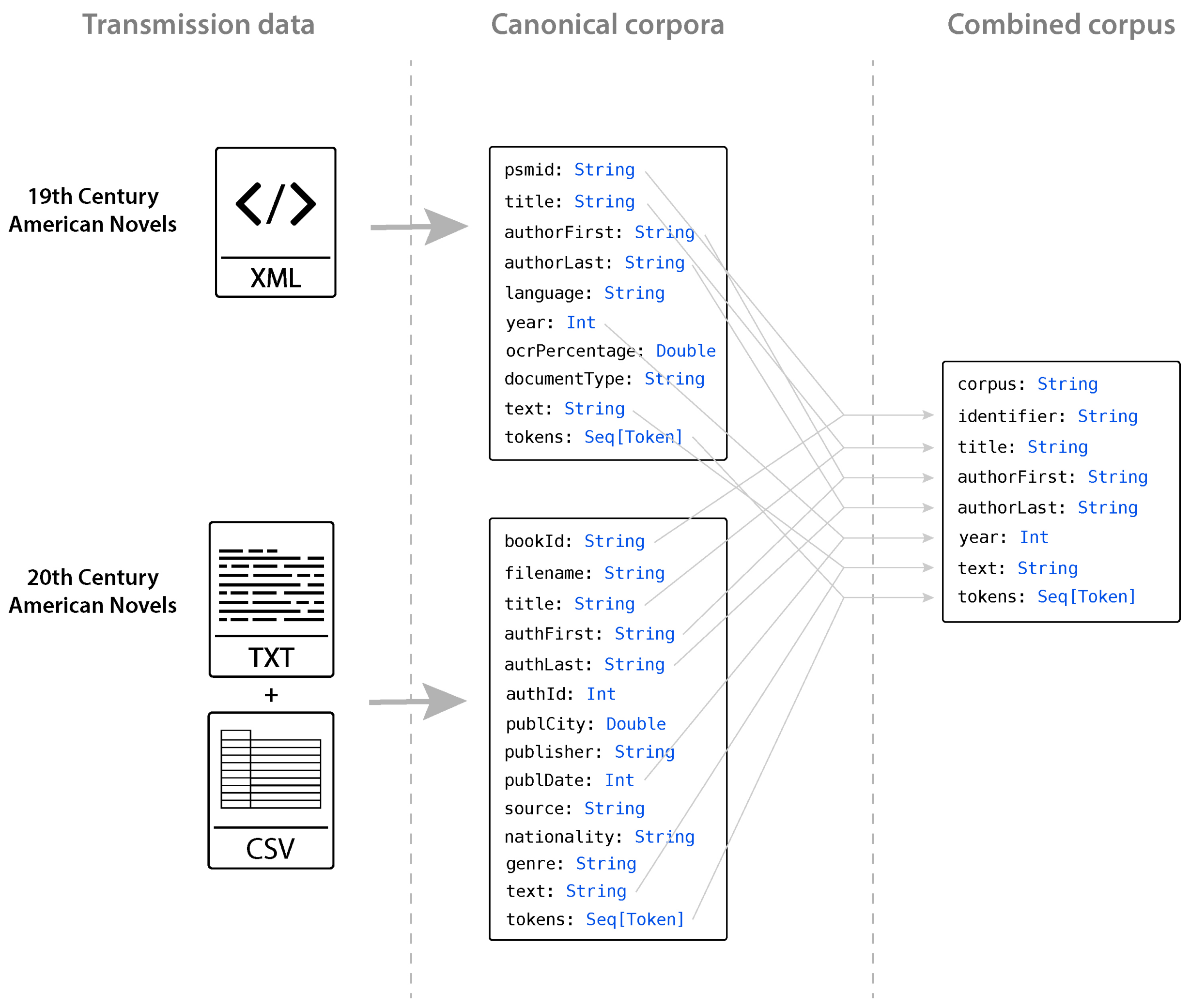

Figure 1: The data pipeline that produces a corpus of American novels spanning the 19th and 20th centuries. The transmission data for each input corpus is extracted into separate Parquet tables, keeping pristine copies of all the original metadata. These canonical representations of the corpora are then duplicated and merged together into a single "combined" corpus, which serves as the basis for the feature extraction jobs needed for the project at hand.

ages. For one thing it makes it very easy to share, publish, and archive the raw data that sits behind a particular project – the Parquet files can just be packaged up as a tarball and send to collaborators or dropped into an institution repository.

Even more important, though, at the level of research praxis – freezing off self-contained copies of the combined corpora makes it easy to maintain a separation between the code that *produces* the corpora and the code that *analyzes* the corpora. When analysis jobs can be written directly against static, unified datasets – instead of having to assemble input dataframes dynamically at runtime – they can easily be broken away from the corpus management system and structured as ad-hoc, decoupled, independent projects, which makes it easier for groups of researchers to iterate quickly on ideas without stepping on each other's toes in a single codebase. Meanwhile, the corpus management code itself can hew to the Unix philosophy of doing just one thing very well and focus exclusively on the task of accessioning and provisioning corpora, without getting cluttered up by analysis code. Research projects can be structured as sets of small, horizontally-scalable modules that interact with the self-contained datasets produced by the corpus manager.

That said, duplicating data in the combined corpora has obvious downsides, if only in that it puts a theoretical limit on the number of ways that we can mix and match the corpora, given a finite amount of storage. We're currently experimenting with hybrid models that would sidestep the need to duplicate the text data, while retaining the essential elements of this approach – the distinction between canonical and combined corpora, as well as the clean separation between corpus management and corpus analysis.

References

[Leskovec et al.2014] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. 2014. *Mining Massive Datasets*, 2nd edition (v2.1). Cambridge University Press, Cambridge, United Kingdom.

[Tiepmar2016] Jochen Tiepmar. 2016. CTS Text Miner: Text Mining Framework based on the Canonical Text Service Protocol. Proceedings of the 4th Workshop on Challenges in the Management of Large Corpora.

[Evert et al.2015] Stefan Evert and Andrew Hardie. 2015. Ziggurat: A new data model and indexing format for large annotated text corpora Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora.

[Kupietz et al.2014] Marc Kupietz, Harald Lüngen, Piotr Baski, and Cyril Belica 2014. Maximizing the Potential of Very Large Corpora: 50ăYears of Big Language Data at IDS Mannheim Proceedings of the 2nd Workshop on Challenges in the Management of Large Corpora.

[Jakubíek et al.2014] Milo Jakubíek, Adam Kilgarriff, and Pavel Rychlý 2014. Effective Corpus Virtualization Proceedings of the 2nd Workshop on Challenges in the Management of Large Corpora.

[van Uytvanck2010] Dieter van Uytvanck. 2010. CLARIN Short Guide on Virtual Collections. Technical report, CLARIN. http://www.clarin.eu/files/virtual_collections-CLARIN-ShortGuide.pdf