

Minimal Computing for Exploring Indian Poetics

Zahra Rizvi <rs_dot_zrizvisasuke_at_jmi_dot_ac_dot_in>, Department of English, Jamia Millia Islamia 
<https://orcid.org/0000-0001-8874-8731>

Rohan Chauhan <chauhan_dot_rohan01_at_gmail_dot_com>, Department of Modern Indian Languages and Literary Studies, University of Delhi 
<https://orcid.org/0000-0002-3259-6270>

A. Sean Pue <pue_at_msu_dot_edu>, Department of Linguistics, Languages and Cultures, Michigan State University 
<https://orcid.org/0000-0001-8463-8578>

Nishat Zaidi <nzaidi_at_jmi_dot_ac_dot_in>, Department of English, Jamia Millia Islamia 
<https://orcid.org/0000-0001-8017-6185>

Abstract

This paper explores multilingual minimal computing and plain text for Indian literatures. It focuses on our workflow designed to produce multilingual, annotated digital critical editions of Indian-language poetry, and to model, explicate, and visualize their poetics. In the absence of digital scholarly corpora, resources developed by citizen scholars working outside of academia are essential; for our team and audience, this includes free and open source solutions — including optical character recognition tools — developed in other contexts. Modeling formal, metrical, thematic, and rhythmic structures opens up the possibility for computer-assisted scholarly analysis across the variously related languages and literary histories of India, which are usually treated in isolation. Positioning our work as a form of minimal computing, we discuss our workflow as a *jugaad* — a North Indian term for reuse and innovation in the presence of constraints.

India is a multilingual society with hundreds of years of continuous literary traditions in dozens of languages, some stretching as far back as two and a half millennia. We use “India” here not only to refer to the contemporary nation-state but also to the greater Indian subcontinent, taking note of the complex, overlapping territorial histories of the premodern “Hind” or “Hindustan,” the “des” or “desh,” “British India,” and Cold War “South Asia.” Additionally, we acknowledge the interconnectedness of India and its diaspora — the world’s largest — with other regions. For any project on Indian literature(s), it makes more sense to speak of multiple literary traditions instead of one. When we speak of “Indian poetics,” we are actually referring to the longstanding traditions of poetics in what come to be understood in the modern period as distinct literary traditions in multiple languages, such as Hindi, Urdu, Bengali, Marathi, Tamil, and Telugu among many others, which cumulatively stand for “Indian Literature(s).” While the specificities of literary traditions are of interest in themselves, we choose to emphasize commonalities and interactions in various poetic traditions of modern Indian languages by reading them comparatively.

Indian literature, imagined this way, is an enormously broad conception that demands philological skills that exceed the capacity of any single individual. Therefore, for our research project, we started with Hindi and Urdu, two languages with which our larger research group, consisting of students and faculty in India and the United States, generally felt some familiarity. These languages have an entangled yet estranged relationship, and they are the two Indian languages, other than English, that our multilingual team knows the best. It is not unusual in India for a person to know two or three languages well enough to share in these literary traditions partially. Languages such as Punjabi and Bengali have some common sources of vocabulary and generally similar grammars to Hindi and Urdu but are written in different scripts with varying pronunciations. Other languages, such as Malayalam, have, in addition to script and phoneme differences, fully divergent grammars — Dravidian rather than Indo-Aryan — though they also incorporate some common sources, such

as Sanskrit.

Hindi and Urdu share a common ancestry: the North Indian speech of the Delhi area. Literary histories of “Urdu” and “Hindi” name this common ancestor as Hindi/Hindui/Hindavi. But they note, this name could be Persian for “Indian” or any language spoken in India (*Hind*). Even though the literary language around Delhi began to put “undue — and sometimes even almost mindless emphasis on ‘correct’ or ‘standard, sanctioned’ speech in poetry and prose” sometime in the eighteenth century, perhaps due to the growing prestige of Persian, it was the British bias in language policy that eventually led to the separation of the North Indian speech into two separate languages defined along religious lines [Faruqi 2003, 850] [McGregor 2003, 912–957]. The gulf only widened further in the late 19th century, when a Hindi-language movement led to the development of Modern Standard Hindi. The movement fashioned Hindi as the language of Hindus by embracing Sanskrit-derived vocabulary and disparaging “Urdu” and its literature as foreign to South Asia because of its “Persian” language elements and metaphors [King 1994]. Tied primarily to the ambitions of identity formation in the emerging struggles of nationalism, as well as the economic competition that was a consequence of colonial linguistic policy in Colonial North India, Hindi and Urdu were reconceptualized as representing different religious identities [Rai 2001]. That separation saw its highpoint during the partition of British India and continues to reverberate across South Asia today. We believe that the intimate yet fraught nature of the history that these languages share provides a fruitful ground for comparisons.

Our collaborative project between the Department of English at Jamia Millia Islamia and Michigan State University, titled “Digital Apprehensions of Indian Poetics,” is supported by India’s Ministry of Education’s Scheme for Promotion of Academic and Research Collaboration (SPARC), which aims to facilitate academic and research collaboration between higher education institutes in India and abroad. This new collaboration facilitates data-intensive textual studies in Indian languages by creating machine-readable, genre-specific corpora and by curating datasets of annotations. We attend to the specific conditions of Indian and South Asian languages, for which there are few scholarly digital editions and where access to computing is often minimal, even at research institutes and universities [Shanmugapriya and Menon 2020]. Our goal is to create digital editions based on minimal computing principles for use by scholars, the public, and students and teachers in the classroom, as well as for ingestion in information systems. This involves creating corpora using optical character recognition (OCR) and adapting existing digital resources, which include those developed by passionate individuals based outside of the academy, or citizen scholars. Therefore, we aim to facilitate the study of texts by the general public, as well as professional scholars. Additionally, we aim to build a vocabulary to document, interpret, analyze, and visualize these textual corpora using linked open data, visualizations, and keywords. Throughout our work, we presume a multilingual audience and aim to facilitate digital humanities research across Indian languages.

In what follows, we expound on our desired outcomes, approach, and architecture. As noted above, we aim to develop digital critical editions and datasets of annotations, including poetic keywords. We approach corpora development as a form of making rooted in the principle of *jugaad*, a North Indian term for reuse and innovation in the presence of constraints. We conclude with a description of the minimal computing architecture that we have adopted for our publishing, editing, and annotation work. We present an architecture that can be accessed across Indian languages — starting with Hindi, Urdu, and English — using minimal computational resources. Finally, we conclude with a note on plain text and the promise it holds for data-intensive textual studies in Indian languages.

Critical Editions and Keywords for Indian Poetics

Critical editions have long been a central component of traditional humanities, primarily for framing our “perception of history, literature, art, thinking, language” by establishing “reliable sources for research” and authorizing and canonizing “certain readings” [Sahle 2016, 19]. Digital critical editions add a new suite of possibilities, including “interactivity, multimedia, hypertext, and immaterial and highly dynamic (or fluctuating) ways of representing content,” which are absent from printed critical editions [Hillesund et al. 2017, 122]. The digital paradigm renders the static text of a printed critical edition into a “laboratory where the user is invited to work with the text more actively,” with the help of integrated features and tools “allowing for customization, personalization, manipulation and contribution” [Sahle 2016, 30]. Harnessing the networked nature of computation, we envision developing a digital platform for scholarly collaboration in publishing. Our goal is to curate and annotate critical editions of individual poets’ works, while providing critical

bibliographies to facilitate research and appreciation.

Through such digital critical editions, our aim is not merely to “bring the past into the future,” but also to furnish it for newer modes of computational inquiries [Hillesund et al. 2017, 123]. Remediating the text not only breathes life into discussions of Indian poetics but also makes the works of a variety of poets more easily accessible to scholars and readers alike. More than that, the endeavor also enables reconsideration of questions about the very notion of the digital text and what it means to read digitally within the context of South Asian languages and literature(s).

We draw from Raymond Williams’ method of “developing accounts of words as reflective essays” for Indian poetics in order to make sense of our textual corpus [The Keywords Project 2020]. Through a mix of distant and close reading approaches on a textual corpus derived through OCR, we hope to develop a historically informed vocabulary of poetic terms and genres that are crucial for understanding the technical field of poetry in Indian languages. We extend Williams’ keyword approach to take stock of the emergent vocabulary that continues to displace established frameworks in the humanities, especially as a consequence of shifting media paradigms. Our objective with the fundamental vocabulary of poetic traditions is not to focus on “fixing (their) definition”; rather, like Williams, we hope to explore the “complex uses” of a variety of conceptual categories to convey the contested nature of their meanings as clearly and succinctly as possible [Bennett 2005, xvii]. When we speak of a fundamental vocabulary, we are obviously presupposing the existence of a canon, and our plan is to foreground the basic implications of the accepted canon in respected literary traditions as much as it is to explore the marginal.

While we want to make this resource accessible to a general audience, we are also keen on designing it for use by specialists and teachers and students in the classroom, as well as in computational or traditional research, by providing a snapshot of the historical evolution of the semantic fields associated with these terms. While the historical variations of meaning might not be as evident for topics pertaining to prosody, we believe this might be particularly useful for pinning down both the “history of words and the contestation of their meanings” for technical poetic vocabularies in different literary traditions over generations, eras, and epochs, acknowledging their discontinuities, ruptures, erasures, and reconstructions, especially in moments of political and social upheaval [The Keywords Project 2020].^[1]

Making Digital Corpora for Indian Languages

Because of the absence of existing Indian language digital corpora, especially for literary texts, “making” became a necessary component of our research. Making has undoubtedly been central to digital humanities and, as recent debates on the issue have clarified, doesn’t necessarily have to begin “from scratch” but instead can start “in media res” [Sayers 2017, 11]. This conception of making — rooted in collaboration, sharing, sustainability, and improvising upon what already exists while critically attending to the specificities of the localities where such making is taken up — focuses on maintaining or remaking instead of reinventing the wheel. Such “critical making,” in Matt Ratto’s sense of the term, not only opens the disciplinary boundaries of digital humanities for introspection but also enriches it by drawing upon practices unfamiliar in its predominantly Anglo-American roots.

For us, critical making also entails localization, whereby we hope to develop, reuse, and repurpose open-source tools and technologies for the particularities of Indian languages and scripts. Our conception of making is rooted in *jugaad*, an intrinsically Indian way of making do, as Pankaj Sekhsaria describes, by “reconfiguring materialities” of existing tools to “overcome obstacles and find solutions” [Sekhsaria 2013, 1153]. Our conception of *jugaad*, which leans less towards engineering than bricolage, is intrinsically tied to localization, and because of this, we inhabit a position closer to a bricoleur than to an engineer. We see our labor as bricolage, in the sense that Claude Lévi-Strauss articulates: using what’s at hand, we design new tools by redeploying a finite set of heterogeneous tools in contexts for which they were not originally designed, in turn not only extending the intended applications of these tools, but also renewing or enriching the stock of tools for future use [Levi-Strauss 1974]. For Sekhsaria, *jugaad* underlies every conversation on technological innovation in India, “particularly north of the Vindhya Mountain Range,” which refers to the territories associated with the Indo-Aryan languages in which the word occurs, and testifies to the prevalence of this word in a large Indian populace, given that the Vindhya Mountain Range essentially splits the country in half [Sekhsaria 2013, 1153]. Analyzing *jugaad* as a “techno-myth” — alongside other cultural practices of making in the Global South

that Ernesto Oroza would deem “technological disobedience” [Gil 2016] — Kat Braybrooke and Tim Jordan contrast the element of necessity in *jugaad* with the element of leisure and choice in the maker movement in the West [Braybrooke and Jordan 2017, 30–31]. This notion of necessity in *jugaad* is also central to Padmini Ray Murray and Chris Hand’s account of making culture in Indian digital humanities, wherein they advocate for a critical dialogue between academic practices and local modes of making [Murray and Hand 2015]. *Jugaad*, then, as a quintessential Indian practice, is primarily to make-do in response to resource constraints. To borrow from Oroza, “misery is not an alternative” for us, as we are motivated to engage with “hybrid” means to achieve our scholarly objectives [Gil 2016].

For example, our initial efforts to localize open-source OCR tools for Indian languages take advantage of two ongoing development efforts in France and Germany. First, we focus on creating ground truth corpora to provide accurate transcriptions for training and testing of both automatic text recognition (ATR) and automatic layout analysis (ALA) models for historical documents in Indian languages using eScriptorium. eScriptorium is an open-source tool, currently under development for hand-written text recognition (HTR) at École Pratique des Hautes Études, Université Paris Science et Lettres (EPHE – PSL), that adapts well for bidirectional scripts. It enables easy application of state-of-the-art neural networks for transcribing and annotating historical documents using an intuitive graphical user interface in modern web browsers. Developed as part of the Scripta project^[2], it integrates the Kraken engine for OCR/HTR^[3], and can be installed both locally and as preconfigured Docker images [Stokes et al. 2021] [Kiessling 2019].

We also draw from ongoing developments within the context of the DFG funded OCR-D project^[4] [Neudecker et al. 2019]. Although it is designed for a different context, OCR-D’s clear guidelines for transcribing and labelling ground truth in the PAGE-XML format offer us a broader framework to maintain consistency across our ground truth. More importantly, we rely on the modularity of OCR-D’s software stack for streamlining workflows best suited for our documents. OCR-D maintains, builds, and repackages a collection of open-source projects in the form of what they call modules. These modules can be painlessly installed with either Makefiles or as pre-built Docker images. Each module comes with an array of tools, which are called processors. These processors can be easily called from the command line for each task in an end-to-end OCR pipeline. Here, image data stored in a workspace or local directories can be easily exposed to either a single processor or several processors sequentially for different processing tasks through their corresponding Metadata Encoding and Transmission Schema (METS) files. Once a task is finished, the information about the processing step(s) is written to the METS file, while the processed data is saved in the respective output directory. This integrated pipeline with pre-configured dependencies becomes extremely salient and functional for the needs of our larger group.

Working with historical documents in different layouts and typefaces, this part of our project will create a publicly available textual corpus in Hindi and Urdu that can be further processed downstream for a host of data-driven humanistic inquiries. These include information extraction methods, such as named entity recognition. Our objective is to work primarily with publicly available digital images accessed through International Image Interoperability Framework (IIIF) API endpoints hosted at different cultural institutions, including the British Library and Internet Archive. Additionally, we will publish well-documented state-of-the-art OCR models in public repositories, along with their corresponding provenance and ground truth datasets, for use in further development of open source OCR for historical print in Hindi and Urdu.

In the absence of robust institutional support, citizen scholars around the world have done much of the work to make Hindi and Urdu texts available on the Internet. Care for languages and poetic experience has led impassioned citizen scholars to launch digital platforms such as Rekhta (Urdu), KavitaKosh (Hindi), and PunjabiKavita (Punjabi). The reach of these projects, especially Rekhta, is enormous, spanning from social media like Twitter, Facebook, and Instagram, to festivals and conferences, to classrooms, to more personal spaces such as family chat groups. Though these platforms provide enormous resources to scholars, their primary and intended audience is not the academy. Our collaboration intervenes here to provide access to these texts as data for an audience that includes the quotidian Indian as well as scholars, poets, information technologists, library systems, and nonhuman agents. Proper metadata is essential for such aims. Using linked open data, we incorporate citizen-scholar projects and resources into our projects as well.

Citizen scholarship is only one example of the changes in the portability of texts through new digital technologies. The

12

13

14

15

16

malleability and flexibility of digital texts, coupled with their easy accessibility and reproducibility, make them increasingly desirable for scholarly inquiry. These affordances motivate our focus on the production of high-quality, accessible digital editions. While learning from community-driven initiatives by citizen-scholars, our objective is to digitally remediate the critical editions of individual poets' corpora in Indian languages, using minimal markup schemes such as Markdown that are easy to reproduce for anyone willing to "decode, receive, and revise" them [Tenen 2017, 21]. The sharing and archiving of these digital editions draw inspiration from projects such as OpenArabicPE and OpenITI. These projects not only version plain-text data using Git but also link it to facsimiles, wherever available, with minimal markup so that scholars can review, contribute, and track changes.

As we will describe below, we follow this model of using plain text in human- and machine-readable Markdown files. While the interface we use to access this data may change, the form of the data, a simple text file, is remarkably versatile. The transformation from Markdown to Textual Encoding Initiative (TEI) XML, too, can be a simple and automatable procedure. However, we require additional layers of annotation, especially for poetry, as well as a multilingual interface.

17

Minimal Computing Approaches to Indian Digital Texts

Prefaced by traditions of localization and open-source activism within India, such as those of Delhi's Sarai program^[5] at the Centre for the Study of Developing Societies, we grapple with questions of software translation and localization of linguistic resources. Our focus is on plain text and the need for an interface that supports both the conceptual and practical dimensions of studying poetics in Indian languages, especially when the digital text itself, no matter how plain, is insufficient. Our Hindi and Urdu texts span alphabets and scripts — left-to-right and right-to-left, Indic, Perso-Arabic, Roman, Devanagari, and unwritten — and yet are mutually aurally comprehensible, for the most part. While we are working under constraints, we dare to dream big and use computing to get around barriers that otherwise look imposing.

18

We use Git to organize and collaboratively write and contribute to our project, as it makes both minimal dependence and minimal maintenance possible. Though used primarily to version computer code, we use Git both in software development and in our other collaborative efforts, including writing. It allows us to avoid dependency on research computing professionals or buying expensive, high-maintenance technology. Minimal computing helps us avoid the alienation of users or the fetishization of tools. While a certain amount of code goes into any GitHub or GitLab project, it makes entry into project development possible and accessible for beginners.^[6] As humanists, the "making" of Git-based projects appeals to our sensibilities. It is useful not only for not learning but also for tinkering around in a perfect *jugaad* fashion, using code and tools in ways that they might not have been intended for before, allowing for backtracking and trying out new branches. Git makes minimal automation possible by circumventing reliance on dedicated web servers; instead, writing and editing work can be done in one's local system and then committed to the central GitHub repository. This feature is particularly useful in situations where Internet access is unreliable or during power outages. Also, it uses low bandwidth, as on each update only the changes are transferred.

19

As noted, we turn to adaptation and localization of existing tools for our collaborative writing workflow instead of reinventing the wheel. Specifically, we use Jupyter Book, part of the Executable Book Project, which enables users to assemble books and articles from a combination of Markdown files, executable Jupyter Notebooks and other sources using Sphinx, the robust documentation engine developed for Python. Though originally used for collaborative computing in STEM fields, we find the executable notebook and book approach of great interest for our digital humanities work, as it allows code to be embedded alongside text. Sharing the source files allows for one team member's work to be reviewed by others, permitting open access to any calculations and visualizations as well as the techniques used to render them. By utilizing Sphinx, Jupyter Book allows these various documents to be easily combined and cross-referenced. Like with Markdown, the text can be converted into HTML for website viewing, as well as into other formats.

20

In our workflow, we had to make some adjustments to attend to the specific requirements of Indian languages and of our multilingual audiences, as well as to the needs of our larger research group. In Unicode, the standard system used to digitally encode the world's writing systems, only the orthographic information about a word can be encoded. Unlike

21

in English, there can be more than one way to write certain ligatures in both Devanagari and Nastaliq scripts, which therefore requires some normalization. Transliteration between scripts, metrical analysis, lexicography, and interfacing with information systems, however, requires additional layers. For example, there is simply not enough information outside of context to distinguish between ڪ (kiyaa: done) and ڪ (kyaa: what). For our larger group, another concern is getting their computer systems ready for the minimal computing toolkit. Though the technical requirements are few, minimal computing still requires some training and setup. Running Jupyter Book locally, for example, requires installing a number of programming languages and libraries to be installed: Python, to run the documentation-generator Sphinx, and Pandoc, a Haskell library, to generate bibliographic citations. We wanted to have the option of two different approaches: 1) to be able to sync from a locally running machine to a remote Git repository and 2) to have an alternative to syncing between a locally running server and the remote.

To address these issues, we took advantage of two newer developments: the JAMstack and using Git as a content management system.^[7] In the JAMstack, web pages are typically pre-generated and written to disk through a process referred to as “Server-Side Rendering” (SSR). SSR is advantageous from a search engine optimization perspective. As the web crawlers of Internet search engines visit a page, they learn not only that there's a page at a given address but also what its contents are. Another advantage is that pages can be loaded very quickly because the pages are pre-generated and static. Finally, these websites are usually relatively small and can often be hosted for free on GitHub, GitLab, Netlify, or other web providers.

While there are several popular JAMstack frameworks, all of which could have handled the tasks at hand, we were attracted to the Vue.js JavaScript framework and within it the Gridsome JAMstack framework. Vue.js recommends a “single page component” approach to web design, whereby code, template, and webpage style declarations are all kept together in one file. We decided to embrace this new approach, hoping it would help us get our projects going quickly. Gridsome uses Vue.js and adds additional features, such as a GraphQL (Graph Query Language) interface to query collections. It relies on plugins, which can import data from different sources. Commonly used sources include hidden or public blogs, Drupal websites, and, as in our case described below, a directory of plain text Markdown files. In addition to markup of text, Markdown files can also contain data of nearly any sort in their header. The following example shows a sample header encoded in the YAML format, a human-readable way of storing or transmitting data:

```
--  
title: Bol  
author: Faiz Ahmed Faiz  
--  
Bol kih lab... (Poem text is here in the body )
```

The sample above includes a “title” and an “author” field. These fields, which usually provide metadata about the text in the “body” of the Markdown file, can be used to store numbers, dates, tags, and nearly any sort of data.

The second trend that we adopted is the move to use Git not only to write or code together but also as a content management system. Here, we used the open-source, JavaScript-based Netlify CMS, which is written using React, an alternative to Vue.js. Easily added to Gridsome and other frameworks, Netlify CMS allows users to authenticate with a Git repository — we used GitLab — and make and commit changes to the repository via a web-accessible editor page. A plugin to Gridsome adds a route to a webpage where users can edit the Markdown header fields online according to a configuration that they specify; we can determine what their content should be (e.g., dates, strings, numbers, lists of strings, or references to other nodes). Netlify CMS then handles the updates to the Git repository. As a result, we are able to have people access and update the data without requiring them to install the full JAMStack on their local computer.

We also knew we wanted to have certain content and data be available in Hindi, Urdu, and English. Fortunately, we were able to adapt the internationalization (i18n) features of Netlify CMS to work with those of Gridsome. Netlify CMS handles translation of Markdown header and body fields by keeping certain common fields in the default locale — we

chose English (“en”) for this project — and by storing “localized” (translated) fields in other locales, as in this YAML header:

28

```
--  
en:  
  title: Faiz Ahmed Faiz  
  bday: 1911.02.13  
  body: Urdu poet  
hi:  
  title: "फैज़ अहमद फैज़"  
  body: उर्दू शायर  
ur:           title: فيض احمد فيض  
  body: اردو شاعر  
--
```

In this sample from an “author” collection, the field “bday” only appears in the default locale (“en”). The fields that can be translated (“title” and “body”) appear also in the Hindi (“hi”) and Urdu (“ur”) locales. Note that the “body” field, which followed the header in the previous Markdown header example, is now contained as a field within the header itself.

29

The Markdown file of a text can then reference its author(s) using the “author” field:

30

31

```
--  
en:  
  title: Bol  
  author:  
    - Faiz Ahmed Faiz  
  body: |-  
    bol kih lab aazaad hai tere  
    bol zabaa;n ab tak terii hai  
hi:  
  author:  
    - Faiz Ahmed Faiz  
  title: बोल  
  body: |-  
    बोल कि लब आजाद है तेरे  
    बोल जबाँ अब तक तेरी है  
ur:  
  author:  
    - Faiz Ahmed Faiz  
  title: بول  
  body: |-  
    بول کہ لمب آزاد ہے تیرے  
    بول زبان اب تک تیری ہے  
--
```

In this example, we specify that there can be more than one “author,” hence that field contains a list, indicated in YAML by the hyphen. The author is referenced using the Roman version of the poet’s name. This allows for a fully human- and machine-readable version of this document. In this way, the plain text files in the directory of a Git repository are treated as a document-oriented database. (Gridsome, in fact, uses the speedy JavaScript LokiJS database internally). When

32

displayed, fields are presented on a webpage in accordance with the client's chosen locale — Hindi, Urdu, or English.

Through this combination of the JAMstack and a Git repository content management system, we can provide localized access not only to viewers but to our contributors, even if they are on a mobile phone or tablet without proper access to a computer. By using continuous integration — in our case, the running of a script when a commit is made to the Git repository — the website is automatically updated when changes to the content are made, and tests are run to assert that the changes are valid. Updates to the data can also be federated; by adjusting the Netlify CMS settings to use permission levels in Git, some users can automatically make changes while others require approval. These proposed changes can come from the public, too, offering a straight-forward pathway to crowdsourcing.

For web-based annotation, we are developing a custom widget that starts from the transcription of the text in its original script (OCRed, if appropriate) stored in the “body” field of a Markdown file. A “genre” field determines how exactly the text will be treated. In general, we address the location of the individual tokens/words by their coordinates in the Markdown file. This allows us to add multiple layers of annotation. The transcribed words are treated as “phrases” that can contain multiple “words.” A name, for example, can be a phrase, but so can a compound word. (Linguists also prefer to have some features, such as future case markers, separated.) Sentences are stored as a span of coordinates (e.g., for poetry indexed by line group, line, and phrase) after we split the original text between spaces, punctuation, and paragraphs. While editing, changes to a custom widget are mapped to a model representation of the text in the web browser and then written to disk following updates.

We are also able to produce a view of the text in the Textual Encoding Initiative (TEI) XML, which is widely used in digital humanities. Sentences map to TEI’s “`<s>`” element, phrases to the “`<phr>`” element, and words are its “`<w>`” element. In this way, we avoid the awkwardness of dealing with right-to-left text in XML editors. Individual words or phrases, moreover, can have additional views or links attached to them, such as scholarly or library-system transliteration or the International Phonetic Alphabet. Also, we can offer views of the individual sentences using the CoNLL-UL format used by Universal Dependencies (UD), a framework for grammatical annotation, allowing us to take advantage of the rich set of annotation tools developed for UD.

For the purposes of developing an annotated corpus, this *jugaad* — using the JAMstack and Git as a CMS — is sufficient. Our plain-text data is relatively small, and it can be read by humans and machines alike. We avoid hosting large image files by instead providing IIIF links to allow transcribers and developers to query the images. Using Git, changes can be tracked and undone. There are few limitations as to the data or links we can provide. For example, interfacing with citizen scholar projects or Wikidata merely requires adding additional fields. For interoperability, the Markdown header field can also approximate (to a certain extent) the graph-based Research Description Framework (RDF), which uses a subject-predicate-object (S-P-O) approach. For example, the previous example can be transformed to RDF as: poem “*Bol*” (S) “hasAuthor” (P) “Faiz Ahmed Faiz” (O). The “document” or “node” serves as a “subject”; the metadata field (e.g., “author:”) as the RDF “predicate” (e.g., “hasAuthor”); and the value (e.g., “Faiz Ahmed Faiz”) as the “object.” In Gridsome, the entire network graph of relations is available both through the query language GraphQL and as a database. Finally, the whole system can be accessed and updated either on a local machine that has the JAMstack implemented or through a web interface to Git using Netlify CMS. While the former is especially useful for those of us doing computational analysis, the latter allows updates to be made from any phone. The data can be easily versioned and progressively archived from Git to Zenodo or other repositories, and the interface expands to meet our needs.

Conclusion: Plain Text for Indian Languages

Plain text is not only simple and dynamic in its potential but also allows authors and researchers to have more creative and scholarly freedom over their work that proprietary tools and formats often lack. If one hopes to make understanding of Indian poetics sustainable and accessible, such minimal computing approaches “reduce the use of proprietary technologies and paywalls to increase access to content, data, and/or source files” [Sayers 2016]. Our project stresses this approach for its contributors as well as for its future audience and users. With plain text, the authors have a close relationship with their work rather than spending time, money, resources, or effort on the tool. More than that, plain text

33

34

35

36

37

files with minimal markup have a much smaller memory size, which reduces both “ecological cost of storage for digital preservation” and accessibility issues for users in low bandwidth regions [Gil 2015]. It eliminates the underlying complex layer of code which mystifies and alienates making from the maker. Instead, it brings all the elements to the surface offering new possibilities for participation and discovery.

Above we have described a collaboration to enable data-intensive textual study of Indian languages. In the absence of existing digital corpora, especially literary corpora, we have turned to making our own, starting with OCR workflows. To do so, we have embraced a form of innovation under constraint by reusing or repurposing what is at hand, which is commonly referred to in North India as *jugaad*. This process also involves sharing methodological innovations as well as data sets. Beginning with poetry in Hindi and Urdu, we have sought to develop digital critical editions as well as sets of annotations that allow for comparative study across Indian languages. We take a minimal computing approach focused on plain text, producing machine- and human-readable data. We use the JAMstack and Git as a content management system to allow for not only computational approaches to text but also to facilitate access and possibilities for contribution for those without a full computer system. Starting with left-to-right and right-to-left materials, we anticipate this approach can be expanded to other Indian languages as we continue to realize the possibilities of minimal computing for exploring Indian poetics.

38

Notes

[1] For an overview of Urdu literary culture, see [Faruqi 2003]. For Hindi, see [McGregor 2003]. Also see [King 1994], [Dalmia 2010], [Rai 2001], and [Mody 2018].

[2] The Scripta project integrates computational humanities and the disciplinary questions in the human and social sciences with the history and practice of writing across most of human history.

[3] Developed by Benjamin Kiessling, Kraken is an extensively rewritten fork of the OCRopus system. It is easily trainable, and is designed to work with a wide variety of scripts by eliminating implicit assumptions about the writing system.

[4] OCR-D is a multi-institutional collaboration for improving open-source tools for OCR.

[5] Since its inception in 2000, the Sarai program at CSD, Delhi has played an active role in facilitating the localization of computing in several Indian languages including Kannada, Telugu, Odiya, Kashmiri, and Urdu.

[6] There are a number of free, easy-to-use GitHub and Git tutorials available on the Internet which provide dummy projects to work on as a hands-on learning process. Our team made use of Meghan Nelson’s “An Intro to Git and GitHub for Beginners (Tutorial)” and the “Hello World” GitHub Guides. Nelson’s tutorial was used by our team members because it has accessible slides and cheat sheets, with reduced technical language.

[7] “JAM” here refers to JavaScript, API, and Markup; these are the “stack” of tools necessary to get the job done. This web framework uses the ubiquitous programming language JavaScript (or higher-level TypeScript), which runs both in the web browser (“client-side”) and as a local or remote server (“server-side”). In the JAMstack, an external API (“A”) can be accessed in Javascript and used to populate a website with data. Individual content pages, such as blog posts, are usually written in Markdown for the “Markup” (“M”).

Works Cited

Bennett 2005 Bennett et al. “Introduction.” In *New Keywords: A Revised Vocabulary of Culture and Society*, edited by Tony Bennett, Lawrence Grossberg, and Meaghan Morris, xvii–xxvi. Malden, MA: Blackwell Publications, 2005.

Braybrooke and Jordan 2017 Braybrooke, Kat and Tim Jordan. “Genealogy, Culture and Technomyth: Decolonizing Western Information Technologies, from Open Source to the Maker Movement.” *Digital Culture & Society* 3 (2017): 25–46.

Dalmia 2010 Dalmia, Vasudha. *The Nationalization of Hindu Traditions: Bhartendu Harischandra and Nineteenth-Century Banaras*. New Delhi: Permanent Black, 2010.

Faruqi 2003 Faruqi, Shamsur Rahman. “A Long History of Urdu Literary Culture, Part 1: Naming and Placing a Literary Culture.” In *Literary Cultures in History: Reconstructions from South Asia*, edited by Sheldon Pollock, 805–863. Berkeley: University of California Press, 2003.

- Gil 2015** Gil, Alex. "The User, the Learner, and the Machines We Make." *Minimal Computing Working Group to Minimal Computing: A Working Group of GO::DH*, May 21, 2015. <https://go-dh.github.io/mincomp/thoughts/2015/05/21/user-vs-learner/>.
- Gil 2016** Gil, Alex. "Interview with Ernesto Oroza." In *Debates in the Digital Humanities*, edited by Matthew K. Gold and Lauren F. Klein, 184-193. Minneapolis: University of Minnesota Press, 2016.
- Gil 2017** Gil, Alex. "Minimal Computing and the Borders of the New Republic of Letters." Michigan State University Global Digital Humanities Symposium, January 27, 2017. <https://youtu.be/D0sfv7xWNSE>.
- Hillesund et al. 2017** Hillesund, Terje and Claire Bélisle. "What Digital Remediation Does to Critical Editions and Reading Practices." In *Digital Critical Editions*, edited by Daniel Apollon et al., 114-154. Urbana: University of Illinois Press, 2017.
- Kiessling 2019** Kiessling, B. "Kraken – A Universal Text Recognizer for the Humanities." *Digital Humanities Book of Abstracts*, 2019. <https://dev.clariah.nl/files/dh2019/boa/0673.html>.
- King 1994** King, Christopher. *One Language, Two Scripts: The Hindi Movement in Nineteenth Century North India*. New Delhi: Oxford University Press, 1994.
- Levi-Strauss 1974** Levi-Strauss, Claude. *The Savage Mind*, translated by Ernest Gellner. London: Weidenfeld Nicholson, 1974.
- McGregor 2003** McGregor, Stuart. "The Progress of Hindi, Part 1: The Development of a Transregional Idiom." In *Literary Cultures in History: Reconstructions from South Asia*, edited by Sheldon Pollock, 912-957. Berkeley: University of California Press, 2003.
- Mody 2018** Mody, Sujata. *The Making of Modern Hindi: Literary Authority in Colonial North India*. Oxford University Press, 2018.
- Murray and Hand 2015** Murray, Padmini Ray and Chris Hand. "Making Culture: Locating The Digital Humanities India." *Visual Language: The Journal of Visual and Communication Research* 49 (2015): 140-55.
- Neudecker et al. 2019** Neudecker, Clemens, Konstantin Baierer, Maria Federbusch, Matthias Boenig, Kay-Michael Würzner, Volker Hartmann, and Elisa Herrmann. "OCR-D: An End-to-end Open Source OCR Framework for Historical Printed Documents." *DATeCH2019: Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, 53-58. <https://doi.org/10.1145/3322905.3322917>.
- Nigst et al. 2020** Lorenz Nigst, Maxim Romanov, Sarah Bowen Savant, Masoumeh Seydi, and Peter Verkinderen. "OpenITI: a Machine-Readable Corpus of Islamicate Texts." *Zenodo*, February 3, 2020. <https://doi.org/10.5281/zenodo.4075046>.
- Pue 2019** Pue, A. Sean. "Digital Apprehensions of Indian Poetics." SPARC Course, *Jamia Millia Islamia*, 2019.
- Rai 2001** Rai, Alok. *Hindi Nationalism*. Hyderabad: Orient Blackswan, 2001.
- Ratto 2011** Ratto, Matt. "Critical Making: Conceptual and Material Studies in Technology and Social Life." *The Information Society* 27 (2011): 252-260.
- Sahle 2016** Sahle, Patrick. "What is a Scholarly Digital Edition?" In *Digital Scholarly Editing: Theories and Practices*, edited by Matthew James Driscoll and Elena Pierazzo, 19-39. Cambridge: Open Book Publishers, 2016.
- Sayers 2016** Sayers, Jentery. "Minimal Definitions (tl;dr version)." *Minimal Computing Working Group to Minimal Computing: A Working Group of GO::DH*, October 3, 2016. <https://go-dh.github.io/mincomp/thoughts/2016/10/03/tldr>.
- Sayers 2017** Sayers, Jentery. "Introduction: 'I Don't Know About Circuitry.'" In *Making Things and Drawing Boundaries: Experiments in the Digital Humanities*, edited by Jentery Sayers, 1-18. Minneapolis: University of Minnesota Press, 2017.
- Sekhsaria 2013** Sekhsaria, Pankaj. "The Making of an Indigenous STM: Technological Jugaad as a Culture of Innovation in India." In *Shaping Emerging Technologies: Governance, Innovation, Discourse*, edited by Kornelia Konrad et al., 137-152. Berlin: AKA Press, 2013.
- Shanmugapriya and Menon 2020** Shanmugapriya T. and Nirmala Menon. "Infrastructure and Social Interaction: Situated Research Practices in Digital Humanities in India." *Digital Humanities Quarterly* vol. 14.3 (2020). <http://digitalhumanities.org:8081/dhq/vol/14/3/000471/000471.html>.
- Stokes et al. 2021** Stokes, Peter A., Benjamin Kiessling, Daniel Stökel Ben Ezra, Robin Tissot, and El Hassne Gargem. "The eScriptorium VRE for Manuscript Cultures." *Classics@* vol. 18.1 (2021). <https://classics->

at.chs.harvard.edu/classics18-stokes-kiessling-stokl-ben-ezra-tissot-garge.

Tenen 2017 Tenen, Dennis Yi. *Plain Text: The Poetics of Computation*. Sanford, CA: Stanford University Press, 2017.

Tenen and Wythoff 2014 Tenen, Dennis Yi and Grant Wythoff. "Sustainable Authorship in Plain Text using Pandoc and Markdown." *The Programming Historian* 3, 2014. <https://doi.org/10.46430/phen0041>.

The Keywords Project 2020 The Keywords Project. "What is a 'Keyword'?" *Keywords*. Accessed July 15, 2020. <https://keywords.pitt.edu/whatis.html>.



This work is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License.