Encoding Multilingualism: Technical Affordances of Multilingual Publication from Manuscripts to Unicode and OpenType

QUINN DOMBROWSKI, MANISH GOREGAOKAR, BEN JOENG (YANG), AND ABEERA KAMRAN

Presented with the history of English-language publishing, it is tempting to draw an arc of straightforward progress from the manuscript to movable type to the typewriter to desktop publishing to the internet. This is not to claim that there were no trade-offs between these steps—particularly with regard to the artistic concessions made in the leap from manuscript to type—but in most contexts the aesthetic losses were mitigated by similar affordances in the new medium, and the possibility of a much bigger audience makes for an easily accepted compromise. Other languages have their own historical narratives of the writing-to-digital shift, where the script used for their writing system plays a major role. The calculus of trade-offs for a single language to adopt a new publishing technology becomes more complicated in a multilingual landscape, and even more so when "multilingual" also means "multi-scriptual." The greater the number of scripts that must be juxtaposed, the greater the number of complications—to the point that a publisher in the 21st century faced with a trilingual text with at least one digitally disadvantaged script[1] could well feel tempted to contract the project to a specialist calligrapher and have it done by hand, sending scans off for printing and distribution as a PDF.[2]

In this article, we briefly trace the major technological shifts in publishing—from the manuscript to social media—through the lens of multilingual publishing, and affordances gained and lost through these changes, before focusing on the era of computing

1. Unicode has begun using the terminology "digitally disadvantaged languages" to refer to languages that are poorly supported by software localization, which includes languages whose scripts are not fully covered by the Unicode Standard and/or languages that lack tooling such as fonts, keyboards (including on mobile), date/time/number/currency formatting, spell-check, optimized algorithms for optical character recognition (OCR), and/or open digital corpora for training large language models (LLMs).
2. Modern accessibility standards functionally preclude this workaround, even if a publisher were to indulge in it, as such a publication would not be usable by screen readers without a layer of OCR remediation—and reliable OCR algorithms still do not exist for all scripts.

and then the internet that led to the rise of the Unicode Standard. Despite their recency, these crucial time periods in the history of multilingual text technology are understudied and are documented primarily through corporate reports, newsletters, and other gray literature not widely accessible to researchers. We draw upon the newly available Unicode Collection at Stanford University Libraries for this purpose. We conclude by bringing together our experiences with interfaces, fonts, standards, and the Unicode consortium to demystify the current technical landscape as pertains to multilingual publishing and advocate for greater engagement with Unicode from people actively involved in multilingual publishing efforts.

From Manuscripts to the Computer

"Multilingual publishing" is capacious through its ambiguity, potentially referring to the publication of documents that contain multiple languages, or to publications that may be individually monolingual but collectively represent norms and practices connected with many languages. The latter meaning of "multilingual publishing" is easier to grapple with, given the thousands of years of written attestation for several scripts and their descendants. One can select case studies from different times and places, juxtaposing them and drawing out common threads, and call the result a "history of multilingual publishing." The former meaning, referencing individual documents that contain multiple languages, offers another lens that allows us to examine how the creator(s) grappled with the challenges of working multilingually.[3] In this article, we move between these definitions as needed to illustrate different challenges across the history of text technologies.

The shortcomings of manuscripts come to mind most readily in a primarily digital age: they are time consuming to produce, and—at least prior to lithography—this labor resulted in the creation of only a single copy. They are physical objects that can exist in only one place at one time, unless digitized as an image, which then falls short of digital text in terms of search and retrieval. It is easy to take the constraints of the digital for granted, as our baseline, and fail to consider the affordances manuscripts offer as another framework for multilingual text creation, and one we could aspire to support with our current technologies. What if multilingual writers in digital spaces—from

3. It is impossible to write about multilingual documents, particularly since the age of the printing press, without acknowledging that a great deal of multilingual publishing globally has happened in a religious context. The Vatican, for instance, has a long track record of multilingual publishing, and SIL, a Unicode Associate Member, is a faith-based organization. Likewise, missionaries and their descendents played roles in spreading text technologies throughout the world. In this article we do not address these motivations and dynamics underpinning the adoption of these technologies, focusing instead on the interaction between the technologies themselves and the languages they attempt to represent.

blogs to monographs to social media—could shift between languages and scripts in text as easily as they do so in speech, or effortlessly arrange text within a canvas in ways that made sense semantically and aesthetically? In written manuscripts we see examples of how writers have done this. Some examples include Mark Dickens (2020) examining multilingual manuscript fragments in Syriac, Sogdian, and Uyghur Turkic from the Turfan Oasis in western China; Alpo Honkapohja (2011) looking at Middle English and Latin in a 15th-century medical handbook; and Galina Miškinienė (2009) studying the 18th-century Kitab of Ivan Lutskevich, which is a Lithuanian Tatar manuscript written in Arabic script with a mix of Turkic, Slavic, and Arabic text. Documents from the Parsi Zoroastrian community in India grapple with the challenges of combining left-to-right (Gujarati, the spoken language) and right-to-left (Avestan, the liturgical language) by writing Avestan upside-down relative to Gujarati (Cereti 1996). A Qing dynasty pentaglot dictionary from the end of the 18th century (Chinese, Manchu, Mongolian, Tibetan, Chagatai) involves four writing directions in a single work (Corff 2016). Even a document with unknown language(s) and a novel script such as the famous Voynich manuscript does not pose a technical problem for a human scribe, who can simply write as they see fit.

The shift from handwritten manuscripts to print ushered in tremendous change in the affordances for multilingual publication. While in practice a set of culturally and linguistically contingent norms constrained the creative range of scribes, we find sparks of the manuscript's creative potential around the edges—sometimes literally, in the case of marginalia. Handwriting is vastly flexible, enabling the writer to freely vary size, shape, position, and stylistic flourishes, all within a single word or across a whole document. The text technologies that followed, from woodblock printing and the printing press to the typewriter, varied considerably in their uptake across different linguistic contexts. However, with the exception of lithography, which facilitated scaling up production of multiple copies of handwritten documents (see Perkins 2017 for a discussion of its use for Perso-Arabic printing), they imposed much stricter constraints on the representation of text—constraints that we have since inherited in the age of the computer. The fixed inventories of characters, tied to sets of professionally produced type (printing press) and possibly keyboards (typewriter), set the precedent for the challenges that writers in multiple languages still face in a computing environment. There has been an uptick in recent scholarship focused on the multilingual history of these text technologies, particularly the typewriter, as they were adopted and adapted across the globe (see Deuchar 2023 [Arabic]; Mullaney 2017 [Chinese]; Lindia 2022 [Amharic]). Nonetheless, much work remains to be done on the global documentation of the hacks, workarounds, and problematic compromises that typewriter and printing press advocates undertook in an effort to force non-Latin scripts into text technology models established for the Latin script.

One finds a similar narrative in the early history of multilingual computing: a new technology was designed, assuming that people would be inputting text in English. Kurt Hensch (2005) describes efforts in the 1960s to adapt the IBM 1401 Data Processing System (a popular mainframe computer) for use in Japan, by only partially implementing one of the three writing systems (katakana) used for Japanese. Using the corresponding IBM 1403 printer to print digital text in Thai gives us an egregious example of the contortions the digital has forced upon those who would write in a non-Latin script. Writers would have to undertake complex script layout, using punched cards for data entry, to record "top vowels and tone marks, followed by consonants, followed by bottom vowels" (Hensch 2005, 25). There is non-trivial mental effort required to visually decompose the text you want to write based on the relative vertical location of the characters' component parts and then enter the text in that manner. Imagine, for instance, having to begin writing this sentence with all the dots on the i's, the lines on the t's, the ascending lines on letters like *l*, *b*, and *f*, and all the apostrophes before moving on to the parts of the letters that appear towards the middle of the line and then entering the commas and descending lines. It's difficult to imagine a way of writing more divorced from the fluidity of the handwritten manuscript and a major hurdle in the process of expression, which is the purpose of writing and sharing text to begin with.

The mainframe computer was poorly suited for multilingual text input, but with the following generation of hardware—the word processor—we begin to see greater attention being paid to multilingual publishing as a market whose needs were worth meaningfully addressing.

Word Processing

The transition from electronic typewriters to "word processors" (a dedicated piece of hardware that performed a similar task, with additional functionality) spanned from the 1960s into the 1980s. A 1971 *New York Times* write-up of a trade show labeled "word processing" a "buzz word" for that year's events, though it glossed the term as "use of electronic equivalent, such as typewriters, procedures, and trained personnel to maximize office efficiency" (Smith 1971).[4] One major corporation in the transition from typewriters to stand-alone word processor hardware was Wang Laboratories, which was founded by An Wang in 1951 and filed for bankruptcy in 1992, as its major product had been thoroughly

4. Manufacturers linked "word processing" to social issues of the day, framing it as "the answer to Women's lib advocates' prayers" that will replace the "traditional secretary and give women new administrative roles in business and industry" (Smith 1971).

eclipsed by the more multi-functional personal computer. Despite Wang's background as a Chinese immigrant, the word processor was not a device that centered multilingual support. Wang's vision of being an American entrepreneur running an American company meant an unbending attitude about the primacy of English (Kenney 1992).[5]

A 1989 *Popular Mechanics* article waxed rhapsodic about word processors, writing off the personal computer as an intimidating new technology that "scared off many potential buyers with its fearsome notion of 'computing,' " whereas "[w]hat millions wanted was a better typewriter" (Scibilia 1989, 72). As usual in Anglophone tech writing for an Anglophone audience, there is no acknowledgment of the fact that the systems profiled in the article and their affordances, including "competent spelling-checkers, with dictionaries ranging from 50,000 to 77,000 words" (72), are tied to a particular language: English. This is particularly ironic given the article's dramatic introduction: "It might have been an ancient Sumerian scribe, etching his cuneiform onto a clay tablet, who first dreamt of a writing engine that would take the drudgery out of moving thoughts from mind to a more permanent medium" (71). In fact, it would take another 15 years for a Unicode proposal to be submitted for representing the Sumero-Akkadian Cuneiform script in a standard way digitally (Everson, Feuerherm, and Tinney 2004). Prior to that, Sumerian scholars would often transcribe characters by hand or use a custom font.

Word processor hardware was poised to be the successor to the typewriter, representing an opportunity for small companies to address linguistic needs and expectations that had remained unmet due to the lack of an adequately large market to attract attention from large corporations. This also had downsides: a 1988 workshop write-up noted that at the small companies manufacturing these niche systems, "changes in personnel, or in the status of companies could lead to software and hardware being suddenly withdrawn—for example the Ferranti scholar. Some computer systems were thought to be somewhat crude by some users, lacking the friendliness of more mainstream software" (Tyler 1988).

Some companies were more adroit in handling the shift from stand-alone hardware to PC add-ons. Beginning in 1988, Applied Electro-Magnetics sold a stand-alone Indian Scripts Video Terminal with two working modes: English-only and multilingual. The unique Indic script features were later repackaged for use with an existing PC as a "video co-processor plugin card which . . . enables you to interact in any Indian language or script at the touch of a key," fully compatible with MS-DOS and text-oriented applications such as word processing software and with a port for plugging in a dot matrix printer. The card could also format the scripts, including bold, italics, shadow, and underlining.

5. In one anecdote from Charles Kenney's book which charts the rise and fall of the company, "Edward Lesnick, who worked in various capacities at the company for many years, recalls that when new Chinese employees joined the firm, they would often speak their native language to the boss, and he would ignore them. He would say, 'This is American company and we speak English here' " (1992, 32).

It implemented the Indian Standard Code for Information Interchange (ISCII) standard endorsed by the Indian government (discussed below), and language support was sold on a per-module basis with each module supporting English Latin script, Devanagari, and one other "regional script" (Applied Electro-Magnetics, n.d.).[6]

Anglophone trade magazines and newsletters from the late 1980s point to a brief-lived flourishing of word processing software designed for multilingual needs, although not all packages were developed with both user experience (UX) expertise and specialist linguistic knowledge.[7] Users also faced the frustration of packages only working on particular hardware. The hardware of greatest interest among multilingual computer users at a 1988 workshop was PC-compatible desktops by a large margin, followed by the Apple Macintosh and Amstrad PCW. However, the LocoScript 2 software which boasted support "in every European language! And soon in Arabic, Farsi and Urdu too" was only available for the Amstrad PCW (LocoScript 1988), as was Qalam (1988), a text system for South Asian scripts. INTEXT (1988) for the PC supported several Western European languages, along with Russian, Hebrew, and Arabic. The fact that not all software ads in these newsletters specified hardware compatibility added to users' confusion; for instance, an ad for the ARROW Publishing System claimed to be "capable of producing countless foreign accents, characters, and symbols commonly encountered in foreign languages" (Arrow 1988). Nonetheless, these freewheeling years saw some degree of font-based support for a tremendous range of both modern and historic scripts, including Mayan hieroglyphs and Glagolitic (the earliest Slavic script). These years saw a less robust computing culture in the USSR, but the rollout of a mandatory computer literacy course on "Principles of Information Science and Computers" caught the attention of Western education researchers (Judy and Lommel 1986).

The high development costs and minimal market for massively multilingual word processing systems had largely kept major tech companies from engaging, but this changed with Xerox's 1984 multi-million dollar contract with Voice of America (VOA) for a version of their Star workstation that could support other languages (such as a 10,000-character Chinese implementation and Arabic), including physical keyboards (Xerox 1985).[8] While the VOA's parent organization in the US government had ini-

6. "Regional scripts" made available through this card were Assamese, Bengali, "Gugarati," Gurmukhi, Kannada, Malayalam, Oriya, Tamil, and Telugu.

7. As noted in a 1988 conference write-up, "Potential users who are fluent in that particular language should be involved in developing and updating systems as well as technical specialists. The view was put forward time and time again; people found that some scripts were good but the word processing facilities were clumsy, while others liked the word processing facilities of particular packages but were unhappy with the quality and accuracy of the actual scripts" (Burke and Ticher 1988).

8. The contract included a list of 20 mandatory languages and 19 non-mandatory languages, some of which—including Chinese—Xerox intended to deliver. Notable mandatory languages included Russian and Ukrainian, Arabic, Farsi, Dari, Pashto, Urdu, Georgian, Bengali, Khmer, Vietnamese, Indonesian, and Lao. The non-mandatory languages included Armenian, Amharic, Hindi, Burmese, and Thai.

tially planned for a English-only computer implementation, Chris Kern and a colleague persuaded management of the importance of multilingual support, leading to a multi-year partnership with Xerox, the first time a company had attempted to support such a diverse range of scripts (Kern 2007). Xerox had already positioned its Star workstation as a leader in multilingual computing on the basis of its support for several Western European languages, Cyrillic, and Japanese (Xerox, n.d.)

Market consolidation happened quickly in the early 1990s. By 1994, a set of recommendations around "Practical Choices for Hardware and Software" for linguists and translators only mentioned WordPerfect (PC) and Word (PC and Mac) among its recommendations, although not without caveats.[9] The author looked to Windows 95 optimistically, which was slated to be released in 27 languages with more to follow (Ball 1994). Even as Word was positioned as a leading option in the mid-1990s, its support for right-to-left languages remained thoroughly broken until 2016 (Stanton 2021). Even better-supported scripts such as Japanese were the result of developing a fundamentally separate version of Word for that script, at the cost of significant developer time and delays, until Microsoft adopted Unicode, discussed below.

The Need for Encoding Standards

Using text, even monolingual text, on computers requires the computer to have a model of how to represent that text in computer memory and how to display that text to the user. A choice of mapping between in-memory representation and abstract text is known as an "encoding," and it gets displayed using a "digital font." Early encodings include ASCII and Extended Binary-Coded Decimal Interchange Code (EBCDIC), which primarily represented the basic unaccented Latin alphabet, as well as JIS X 0201 for Japanese.

As more and more documents were being composed on computers, people developed more encodings for their usage, occasionally publishing them with fonts (see Rockwell 1986). For personal word processing, picking an encoding/font for a document would be sufficient, especially if the end product was printed. However, the moment one needed to share documents in a digital format, or write software that displays text, the sender and receiver of the document or software would need to agree on how their computers would interpret the text. To facilitate this, more effort was put into standardizing *encodings*, conventions which could be made available on most computer systems and selected as needed. This was a better solution than the common

9. Peter Ball (1994) notes that Word supports both Western and European languages, but "the support for languages outside the standard 12 [languages included in the Western European version of Windows: Danish, Dutch, English, Finnish, French, German, Icelandic, Italian, Norwegian, Portuguese, Spanish, and Swedish] is far from perfect."

workaround—shared custom fonts that mapped other characters to the standard Latin script letters—offered.

The custom font approach was quite functional within scholarly communities, in which a small group of people could agree on an informal standard that text in a given script would be written in a particular font, and that font would be shared throughout the network, passed between professors and students, and among students, on diskettes. For a language such as ancient Sumerian, where text in that language is produced by scholars for other scholars, this approach is adequate. But for living languages, there are considerations beyond simply producing text for an in-group. Ideally, the language should be incorporated into software interfaces. Writers in a language community have different aesthetic needs and desires that call for the use of multiple different fonts. Standards are essential here so that changing the font will not change the text. A language community that can freely use a wide range of digital fonts for different expressive purposes is on its way to reclaiming some of the graphic flexibility lost in the shift from manuscripts to the printing press and typewriters.

The need for encoding standards became more pressing with the growth of the internet and, later, the World Wide Web, as more and more people were sharing documents across computers. Systems would typically track the encoding of a document in some form of metadata, either a part of the document format itself or served separately.

Having to choose an encoding was not the only stumbling block for using text on computers. Some writing systems required complicated rendering support to accurately render ligated text, others required transliterating input methods to allow for input of their large character set on regular-sized keyboards, and yet others required careful processing of bidirectional text (Becker 1984). Multilingual documents suffered greatly from the need to agree on encodings: most encodings supported at most two scripts, typically Latin and some other script. If one wished to use characters from more than one encoding, they were typically unable to, unless they were working with a format like RTF that supported switching encodings on the fly.[10] It was common to encounter gibberish, dubbed *mojibake* from a Japanese word meaning "character transformation," resulting from text being read using the wrong encoding, and users would have to manually switch encodings until they found one that made the text legible.[11]

10. For a more detailed discussion of encoding within the context of the entire "text stack"—encompassing coding standards, digital fonts, input methods, and shaping engines—see Anushah Hossain (2024).

11. One famous instance of *mojibake* that circulated on the internet involved a person writing the address their Russian pen pal had emailed them, exactly as it appeared on their screen, without recognizing the *mojibake* that had taken place. Remarkably, Russian postal workers were able to decode the address (presumably by transcribing the handwritten *mojibake* back into a computer and changing the encoding) and deliver it successfully. A photo of this card is available at https://web.archive.org/web/20221220132400/https://commons.wikimedia.org/wiki/File:Card_with_a_cyrillic_address_in_wrong_encoding.jpeg.

Unicode

In 1988, Joseph D. Becker, a computer scientist working at Xerox, put forward a proposal for “an international/multilingual text character encoding system, tentatively called Unicode,” described as a “ ‘wide-body ASCII’ that has been stretched to 16 bits [from 7, supporting only 128 characters] to encompass all the characters of all the world’s living languages” (Becker 1988).[12] It was the culmination of work begun in 1987 at a meeting between Apple and Xerox.

The financial logic of large corporations around priorities is visible in this initial Unicode proposal; to make a good case for the encoding, it had to be economically viable and not only a service to the world. Section 1.2 of the overview includes a table that “ranks the world’s writing systems roughly in order of commercial importance, as measured by the total GNP of countries using each system” (Becker 1988). Latin is pegged at 68% of the world GNP, CJK and Cyrillic at 14% each, Arabic at 3%, “Devanagari family” (North Indian Brahmic scripts) and Korean at 1%, and the remaining are negligible. The list is hardly comprehensive, but the point is clear: a cost-benefit analysis would not work out in favor of supporting most of the world’s scripts on a one-by-one basis. The technical structure of Unicode encoding introduced the possibility of a different calculus: implementing Unicode would lay significant technical groundwork for making *any* living language work.

This initial proposal described Unicode as a “multilingual generalization of ASCII codes,” where “each individual Unicode code is an absolute and unambiguous assignment of a 16-bit number to a distinct character [in one of the world’s writing systems]” (Becker 1988). It lays out five major principles for the system:

- Drawing a distinction between *characters* (“abstract text content-bearing entities”) and *glyphs* (“visible graphic forms”).
- Unifying “tens of thousands of equivalent ideographs” that may look different as used in China, Japan, and Korea; this *Han unification* Unicode initiative quickly led to a dedicated sub-committee and no small amount of controversy.
- Public vs. private: an area of the Unicode code space would be set aside and not used for assigning characters from the world’s writing system. This is important to support experimentation, vendor-specific implementations, and some of the flexibility that users enjoyed through custom fonts prior to the introduction of Unicode.

12. The adjective *living*, here, is noteworthy, as it excludes an intrepid group of computer users who had at that point made strides in digital publishing through the use of custom fonts: scholars of dead languages written in historical scripts. Becker addressed this audience briefly in the document, designating their needs as more suitable to an unassigned “private use area,” recognizing their strategy of custom fonts as legitimate and providing space for it such that it would not interfere with modern text.

- Plain vs. fancy text: plain text is defined as purely a sequence of Unicode, in contrast to “fancy text” which contains more information in some manner.
- Process-based design: the Unicode proposal frames text encoding as “exist[ing] solely to support the various processes that act upon text,” including “rendering, filtering, and so on.”

Unicode vs. ISO 10646: 16 bits or 32 bits?

The first Unicode document takes a strong stance on the adequacy of 16-bit encoding.[13] While this may seem like a niche technical detail, it has both philosophical and practical consequences. Philosophically, it claims that a “reasonable” and moreover “proper” notion of a “character” can be formulated such that all the modern world’s writing can be usefully encoded without needing more than 65,536 of them. Practically, this meant that some scripts would have to be handled in a fundamentally different way than they had been to date. According to one member of what would become the Unicode Technical Committee (UTC), the original plan was to use decomposed combining “jamo” characters for Hangul, not encode any Arabic positional forms, and potentially even use something like Ideographic Description Characters for composing uncommon Han characters.[14] This would have significant consequences for users of those writing systems, and while the implementation of Unicode did not go quite so far, the “Han unification” working group designed to reduce the number of characters needing representation rankled (Searle 2004).[15] However, practical considerations led to scrapping the more extreme plans for Hangul. According to the UTC member, “Many systems’ font technology hadn’t advanced to the point where it could handle producing a single glyph from multiple characters, which forced large sets of composite [H]angul, Arabic presentation forms, etc.”

Unicode was not the only effort to implement a multilingual encoding system around this time; in 1989, led by British computer scientist Hugh McGregor Ross, the International Organization for Standardization (ISO) published ISO/IEC DIS 10646 v.

13. One UTC committee member noted the amount of resistance to going beyond 16-bit (or 2-byte) encoding came from a reluctance to double storage requirements and transmission time, which were significant barriers at the time. They explained that an early plan was “wedded to a mechanism that would not allow any control-code bytes, making it ~94x94 possible characters in 2 bytes, limited them to about 36K characters instead of 65K. So there were escape sequences [added] to load different ideographs in part of that space.”
14. Ideographic Description Characters are a Unicode block introduced in 1999 used to describe the layout and subcomponents of CJK ideographs.
15. Steven Searle (2004) claims, “It is impossible to create a digital library or to digitize public records in Japan using the Unicode Basic Multilingual Plane. That’s because there are only a little more than 20,000 Chinese ideographs on this plane, and only 12,000 are applicable for Japanese.” At the same time, it is worth noting that this gap resulted from the fact that the “missing characters” were not part of any extant standard at the time, as Japan’s JIS-C 6226 kanji standard and its successors were all included in the unified Han character set.

1, a proposal for a universal character set that had buy-in from Japanese and European researchers. ISO 10646 could encode up to 679,477,248 characters but was more complex and required more space for storing text. US-based computer manufacturers chose not to implement it, opting instead for the technically simpler Unicode approach. The difficulty of achieving tech company buy-in made the standard, practically speaking, dead on arrival. This situation put enough pressure on national representatives to the ISO that the first draft was scrapped, and ISO/IEC 10646 v. 2 was reinvented wholesale on the basis of Unicode in 1991 (Dvorak 1994, quoted in Searle 2004). To this day, the Unicode website includes an entire section of the FAQ dedicated to ISO 10646.[16]

While computer manufacturers pointed to the complexity of 32-bit ISO 10646 as a reason it was impractical to implement, even with Han unification underway, Unicode soon reconsidered the feasibility of 16 bits.[17] In the development of Unicode version 2.0 (released in July 1996), the standard was modified to use a 32-bit encoding by creating a set of 16-bit planes, one of which would contain all characters previously encoded in the original 16-bit space. This facilitated a long ramp-up for migration to 32 bits, since it took more than a decade for the number of allocated characters to pass that earlier threshold. As of Unicode 15.1, the Unicode Standard has assigned 149,813 code points to characters; 97,680 are Han characters.

"Do You Have an ISO Standard?"

Beginning in 1988, Lee Collins began development of a database of Unicode characters, drawing on an East Asian Coded Character Set (ANSI/NISO Z39.64, known as EACC). The EACC includes 13,479 Han characters, in addition to Japanese kana and 2,028 Korean Hangul, the latter being mostly composite characters that the initial vision for Unicode had hoped to avoid. The early Unicode implementation working group compromised on composite characters in other contexts too. To facilitate "round-tripping" (in which a text in a different encoding standard is converted to Unicode and then back to the original standard, without loss), composite characters that already existed in an ISO standard were added to Unicode. This push-and-pull of one standard using another to justify its decisions (and later vice-versa) continued between Unicode and ISO standards for some time, leading to a large number of carve-outs and exceptions for characters used in Western European languages that could reasonably be treated as "composite" and not requiring a separate code point but were granted full-character

16. https://www.unicode.org/faq/unicode_iso.html.
17. The UTC member acknowledged the fact that "the ability to sell it as a 16 bit fixed width encoding was probably part of the reason for its success," which amounted to an unintended bait-and-switch for the tech companies.

status nonetheless.[18] Over time, the prevalence of composite Latin characters grandfathered in due to ISO standards lowered the bar for new additions of the same nature.[19] Because long-term stability is an utmost priority for Unicode, once a character is added to the standard, it will not be removed, meaning there are now multiple ways of writing certain characters leading to specifications of canonical equivalence and normalization.

As Unicode set about including new scripts, the existence of an ISO standard for a script made it more likely that the script would be encoded in Unicode in a way that preserved the existing implementation and enabled round-tripping. Other, non-ISO standards, including ISCII (Indic scripts) and Shift-JIS (the most widely used encoding for Japanese prior to Unicode), were not accommodated for round-tripping or even implemented by Unicode but served as a more general guide.[20] ISCII's model mapped characters from different scripts with the same origin to the same code points, differentiating them with a control character. This was at odds with Unicode's manner of encoding, in which each character in each script got its own set of characters. While round-tripping was originally possible in principle, by the late 1990s there was enough divergence that it failed (Hellingman 1997; see also Agenbroad 1991 for a series of technical complaints about the implementation). An ISO standard (which privileged Latin and Russian Cyrillic script users) played a role not unlike a flag[21] in Eddie Izzard's classic satirical sketch of British imperialism: in the absence of one, the script would be implemented by Unicode as best as possible in consultation with experts and/or a user community, but competing interests and perspectives (e.g., between different governments when a script spans national borders, or groups within a language community with different visions for their script) can lead to conflicts that stall progress on script encoding while issues are sorted out.[22]

18. ISO 8859 was a major target for this work. ISO 8859 is a standard for an 8-bit extended ASCII relevant to Latin-script European languages, as well as the Greek, Cyrillic, Hebrew, Arabic, and Thai scripts.
19. For example, the Romanian national standardization body appealed to Unicode directly about adding Ț/ț, which was included in the Unicode 3.0.0 update of September 1999, which corresponded to ISO/IEC 10646–1:2000.
20. ISCII emerged from efforts at IIT Kanpur in the 1980s to develop computing from the ground up for Indic scripts, due to fundamental differences between abugidas (where characters represent segments or syllables) and alphabets (Sinha 2009). Prioritizing shared character origin as an organizing principle flattened some of the uniqueness of each script.
21. Relatedly, the topic of flag emoji is another contested landscape, as described in Daniel (2022).
22. One of the best-known examples of this is Mongolian, which was encoded using a phonetic model, needing a massive amount of language-specific special-casing to render correctly, none of which had been documented and therefore one could not rely on fonts implementing it correctly. Likewise, the Unicode Collection archive has several documents related to a dispute over adding the letter Қ (used in Turkic languages spoken in post-Soviet nations such as Kazakhstan and Uzbekistan) to the Cyrillic code block, which the Russian Federation opposed as being unnecessary. Following the dissolution of the Soviet Union, Cyrillic became the locus of several disputes over both character encoding and the defined sort order for the characters. The Cabinet of Ministers of newly independent Ukraine sent a letter, dated September 23, 1991, proposing a new script for Unicode, the "Neocyrillic alphabet" that "embraces alphabets of all the six Slav languages" (Komisarenko 1991). The Unicode Collection also includes a letter from the Head of the Informatics Commission of Taras Shevchenko Ukrainian Language Society, dated November 26, 1991, to the convener of subcommittee JTC1/SC2/WG2, arguing in favor of including the Ukrainian Cyrillic letter ґ in the ISO standard. The letter notes that the Unicode project included a "full collection of Ukrainian characters but in chaotic sequences" (Komisarenko 1993).

OpenType and Digital Fonts

Unicode made it technically possible, for the first time since the manuscript, to engage in multilingual publishing that could fairly easily juxtapose any arbitrary set of scripts. The migration to Unicode was a serious technical undertaking, but one with considerable value for phasing out separate software versions with different language support, as reported by Microsoft at the 1999 Unicode conference (Pratley 1999). Script encoding by itself, however, is not enough: fonts are necessary for actually rendering the text on a screen or in print. Furthermore, a font that simply has a glyph for each character in the writing system is not enough: some writing systems require complex ligatures, positional forms, or combining rules that must be implemented in order for the text to be legible, let alone natural looking.[23] To add to the complexity involved in producing legible text in many major scripts, the software used to enter and display the text needs to use a shaping engine to turn the input keystrokes into the right combination of ligatures and forms that exist in the font—or into reasonable fallback options if the optimal form did not exist.

Modern fonts use a set of formats descended from PostScript, developed at Adobe in the 1980s with a focus on high-fidelity print representation of glyphs. Jason E. Lewis and Bruno Nadeau (2009) criticize PostScript and its ubiquitous descendant formats, TrueType and OpenType, for continuing to prioritize print as a paradigm, at the expense of emerging 20th- and 21st-century media including 3D environments.[24] While the most common narrative of OpenType is centered on the "font wars" of the tech industry in the 1980s, Anushah Hossain (2024) offers a nuanced history that centers the impact on what the software industry designates "complex scripts" (namely Arabic and Indic scripts). While script complexity is in the eye of the beholder, the technical "solutions" for addressing the ligatures and variants in these scripts were indeed complex, particularly prior to the debut of OpenType. Two aspects of OpenType's feature set—data tables for glyph positioning and glyph substitution—streamlined what had been an awkward set of workarounds, although adoption across South Asia took time due to delays in the full implementation of Microsoft's Indic shaping engine (Hossain 2024).

With the transition to digital typography, new possibilities also emerged for Nastaliq printing. The first digital Nastaliq typeface, Noori Nastaliq, was developed in 1981 by Monotype and Ahmad Mirza Jamil. Digital typography still operated on some of the assumptions of physical type, including those of the horizontal baseline. To support the

23. The internet is full of examples of "Arabic" translations—not infrequently in official text—written in a way where the letters appear disconnected from one another, as a result of using a combination of font and software that do not support the complex positional variant rules needed for producing correct Arabic text.

24. TrueType was developed by Apple initially as a competitor to Adobe's PostScript. Microsoft, with later contributions from Adobe, extended TrueType as OpenType after failing to license TrueType from Apple (Haralambous 2007).

large variety of contextual letterforms as well as the diagonally stacked nature of words in Nastaliq, Noori Nastaliq required over 20,000 separate glyphs. The majority of these glyphs were ligatures and therefore could not be mapped onto Unicode. The font could only be used on Monotype's Lasercomp and was not available to average users. This meant that despite the breakthrough introduction of a Nastaliq font, the majority of Urdu publishing relied on handwritten lithography until the late 1990s.

The development of OpenType allowed for complex, algorithmic glyph shaping and positioning. For Nastaliq, this meant fonts required far fewer glyphs, relying on the font and layout technology to properly lay out the glyphs vertically within the line.[25] By inputting individual Urdu letters, the user can receive properly laid-out Nastaliq writing with the positioning and shaping of the glyphs handled automatically by the technology.

However, it should be noted that even with OpenType's advancements, the underlying model of digital typography is still highly Latin centric. There is still an assumption of a horizontal baseline, and the technology is in some ways "hacked" to display a writing style that breaks those assumptions. As such, while it is now possible to create attractive Nastaliq typography, the technological skill required to engineer a Nastaliq font is prohibitively high, even for accomplished font engineers. In particular, the placement of nuqtas in a Nastaliq font so that they do not clash with one another and appear balanced within a diagonally stacked word remains a key challenge for font engineers. As such, there are very few Nastaliq fonts available compared to other Arabic script styles such as Naskh or Kufic.

Given the crucial role that fonts play in Unicode text readability, advances in technologies that underpin the representation of text on the internet have come a long way in addressing earlier challenges, such as the need to have a font with the appropriate script coverage installed on the display device. The introduction of the @font-face attribute in CSS2 enabled embedding custom fonts for display on a website, to ensure universal legibility even for text in obscure or historical scripts. Still, this required thoughtful planning on the part of web developers, to identify and embed fonts that together cover all the scripts used in a given web page. Google's OpenType Noto font family has closed this gap by providing open-licensed coverage of most of the scripts encoded in Unicode, in a uniform weight and style for aesthetically pleasing script juxtaposition.[26]

25. Noto Nastaliq Urdu, an OpenType font, contains 1,433 glyphs, far fewer than Noori Nastaliq.

26. The name *Noto* comes from "no tofu," referring to the term for the white square or rectangle character (□/□) that typically appears by default when no font is available to render a Unicode character. As of April 2021, the Noto family covers 95% of the non-CJK characters in Unicode 13, but only 32% of Unicode's CJK coverage (Twardoch 2021). Because the OpenType specification only supports up to 65,536 glyphs (and there may be a need for multiple glyphs for a single code point, due to the "complex script" issues described above), Noto is implemented as a family of fonts, one per script, rather than a single mega-font.

Despite minor criticisms of the font quality for historical scripts (which have increasingly been handled as outsourced projects rather than developed in-house), and the legibility when displayed at a small size on mobile devices (Park 2023), having a relatively comprehensive, uniformly designed cross-script font family is a significant step forward for multilingual publishing, particularly online and in an ecosystem of apps.

Social Media

What is "publishing" in the 2010s and 2020s? The media and device landscape has shifted once again, such that interacting with digital media (and multimedia) is nearly ubiquitous among adults worldwide, but primarily through mobile devices. Where earlier days of the internet saw content spread across a huge swath of diverse websites—including traditional news, lifestyle, and informational publications moving online, as well as blogs that anyone could write and anyone could subscribe to updates from using RSS—control of the content landscape has moved into the hands of a small number of tech companies through the use of mobile apps. On a website, with enough technical skill, an individual creator could make and implement their own decisions about how to display multilingual text, including using embedded custom fonts mapped to the Unicode private use area, if needed. The level of graphical freedom and flexibility that building one's own website affords is remarkable and has been increasing as web standards more broadly evolve to support complex and creative layout. A custom-coded website in the 2020s comes closer to the affordances of handwritten manuscripts than anything that preceded it digitally. However, when publishing takes place in the walled garden of an app, users have little recourse if the app does not support their script due to a lack of fonts (now less common thanks to the Noto font family) or mangles the display (still common, especially for lesser-known "complex scripts" such as Javanese).[27] While a computer with a keyboard and word processing software are still indispensable for people who regularly produce digital text, much casual communication (social media, but also comments on articles, which in some linguistic contexts involve a polyglot discourse that is by itself "multilingual publishing") is done using mobile devices. If a script does not have a reasonably easy-to-use mobile keyboard, it increases the likelihood that users will resort to a simpler text input method—which often looks like transliterating into Latin, or choosing to communicate in another

27. One workaround that some users have resorted to for scripts that are still not in Unicode is custom, script-specific text authoring apps such as Zmongol (https://install.zcodetech.com) that compose the text as an image that can then be downloaded and posted to social media. This approach has several downsides, including lack of searchability for the resulting text and the difficulty of combining this image-based text with arbitrary other scripts: on a website, one can display an image in the middle of a sentence, but social media apps generally place uploaded images before or after text.

language. At the 2023 Face/Interface conference, typographer and typography instructor Hrant Papazian noted that young people are bombarded with images and text on social media associating English with wealth, power, and cultural capital. Cultivating youth pride in their own identity, language, and script—with long-term consequences including language maintenance, particularly in the diaspora—is made much harder when that script is displayed in an unevenly mangled way across operating systems and apps (Papazian 2023; Syauqi 2023).

In a world where publishing is so often mediated by tech companies' apps, we are dependent on those companies to take an interest—either for their own profit-driven market plays or, less common, as a charity project—in investing in better infrastructure for multilingual publishing on their platforms, particularly for scripts with a smaller user base. In practice, however, the choices that companies make around script support cascade down into other choices about things like content moderation. Facebook recognized that many users in Myanmar were not using the Unicode implementation of the Burmese script but rather one traceable back to a free font that mapped Burmese characters onto different code points within the assigned Burmese Unicode space, using a different logic for how to define characters—meaning no one-to-one mapping with the official Unicode Standard was possible (Hotchkiss 2016). Facebook implemented support for this manner of writing Burmese in its app, automatically analyzing character frequencies to determine whether a post was using Zawgyi or Unicode and picking the display font accordingly. As of 2019, when the government declared an official switch to Unicode with a two-year implementation window (AFP-JIJI 2019), around 90% of Burmese internet users used Zawgyi. But it is difficult to celebrate Facebook's accommodation of Zawgyi (see also LaGrow and Pruzan 2019) as a win for multilingual publishing in the context of the genocide of the Rohingya people fueled by hate speech that spread like wildfire through Burmese-language Facebook as a result of inadequate Burmese-language moderation, as documented by Erin Kissane (2023).

Infrastructure Futures for Multilingual Publishing

Multilingual publishing is digital. Even alternative publications such as zines are likely to draw upon digital tools at some point in their production.[28] While work remains to be done, decades of labor on standards bodies and within the tech companies tasked

28. The countercultural, experimental nature of zines makes them a compelling platform for multilingual publication, in contrast to more formal venues in which language choices are more likely to be policed, implicitly or explicitly. Examples include *The Diné Reader: An Anthology of Navajo Literature*, written in "the Diné, Hawaiian, Cree and Spanish language with some intertwined with English" (Lee 2023) and *Canto Cutie* (https://www.cantocutie.com/), bilingual in Cantonese and English.

with implementing those standards means that we finally have technical infrastructure for doing multilingual publishing across almost any set of languages that can, much of the time, simply work.

Who shapes the future of Unicode? Looking at the members list and seeing that nine of the 10 full voting members are large tech companies might cause some alarm, that Silicon Valley is controlling the digital future of the world's scripts. In practice, the corporations active in Unicode are more interested in ensuring that the organization's work *happens*, rather than shaping individual decisions. Scripts and new emoji need to continue to be encoded based on careful review and expert opinion, the Unicode libraries need to be maintained, and the databases need to be up to date. Because the membership dues particularly at the full member level help pay the modest administrative costs of the Unicode consortium, they have recently engaged in outreach to tech companies to encourage them to join—and many have, not least because Unicode runs a project called Unicode Common Locale Data Repository (CLDR) that lowers the bar to software localization.

The difference between voting members (mostly tech companies and a couple government organizations) and non-voting members (a much broader group, including universities, companies outside the United States, and individuals) is mitigated by a very strong culture of consensus and taking in feedback from experts—to the point where voting on anything other than organizational processes almost never happens.[29] There is a broader critique one can make about the risks of relying on corporations for so many pieces of the publishing ecosystem: from the hardware we use to compose and read text; to implementing the text stack of encoding standards, fonts, and shaping engines needed to make that text legible regardless of script; to managing social media platforms so publishing isn't turned into a vehicle for violence. The transformation of Twitter into X following its acquisition by Elon Musk may serve as an object lesson in these risks and a spur to change.

If we can break out of the walled gardens of social media and turn our attention towards creative use of a broader digital publishing stack—including pushing forward with the development of HTML5, CSS, and JavaScript—there are opportunities to imagine a future for digital multilingual publishing that values aesthetics as well as information, reincorporating visual affordances offered by manuscripts that have been sidelined in the evolution of text technologies.

Looking ahead, there is much to be hopeful about: Unicode has moved, and continues to move, towards much greater diversity in gender, background, and life experiences, which will shape the future development of the standard. Likewise, a new generation

29. As one committee member put it, "I heard very early on in my Unicode experience that if something has come to a vote, something very bad has happened."

of typographers and typography researchers—from Ariq Syauqi's work on Javanese and Balinese, to Niteesh Yadav's efforts to address typographic shortcomings in augmented reality/virtual reality (AR/VR) environments, to students of Hrant Papazian and Yara Khoury designing creative, beautiful typefaces for non-Latin scripts—are laying the foundation for beautiful, thoughtfully designed digital multilingual publishing.

Multilingual publishing inherently brings with it challenges not faced by monolingual publishing—even more so if that language is English, as so often it is. But the difficulty gap is smaller than it has been for centuries. The best way to continue to push the standards and technical specifications forward is to use them and report on where they fall short. The standards may feel distant and impersonal, but they are written and maintained and revised by *people* who care deeply about linguistic diversity. Your next moment of technological frustration may be the use case they need to move forward with changes that will make the road easier for the next person.[30]

Acknowledgments

We would like to thank Stanford University Library Special Collections and the Silicon Valley Archives for their stewardship of the Unicode Collection and archivist Chloe Pfendler for processing it for research use.

References

AFP-JIJI. 2019. "Unicode in, Zawgyi out: Modernity Finally Catches up in Myanmar's Digital World." *Japan Times*, September 27, 2019. https://web.archive.org/web/20200822105601/https://www.japantimes.co.jp/news/2019/09/27/business/tech/unicode-in-zawgyi-out-myanmar/.

Agenbroad, James E. 1991. Letter to the secretariat of the Unicode Consortium. Stanford Special Collections M2864 Unicode, box 63, JDB, Indic Standards, 1983–1994, folder 1.

Applied Electro-Magnetics. n.d. "Transcript—the Indian Scripts Card." Advertisement. Stanford Special Collections M2864 Unicode, box 63, JDB, Indic Standards, 1983–1994, folder 1.

Arrow. 1988. "Accents, Symbols & Foreign Characters." Advertisement. *SESAME Bulletin: Language Automation Worldwide*, vol. 2, part 1. 1988. Stanford Special Collections M2864 Unicode, box 72, JDB, Printed Material, 1987–1991, folder 3.

Ball, Peter. 1994. "Practical Choices for Hardware and Software." *Proceedings of Translating and the Computer* 16 (November). https://aclanthology.org/1994.tc-1.10.pdf.

30. While there are other ways to get involved with Unicode, including helping research and evaluate unencoded writing systems and writing proposals (contact the Script Encoding Initiative for that area of work at https://scriptencodinginitiative.github.io/), the best route for reporting issues and problems is through the contact form at https://unicode.org/reporting.html.

Becker, Joseph D. 1984. “Multilingual Word Processing.” *Scientific American* 251, no. 1 (July): 96–107. https://www.jstor.org/stable/24969416.

———. 1988. “Unicode 88.” August 29, 1988. https://unicode.org/history/unicode88.pdf.

Burke, Sally, and Paul Ticher. 1988. “Multi-lingual Word Processing and Voluntary Organisations: Report of a Conference (1st, London, England, June 1988).” Community Information Project, London. https://files.eric.ed.gov/fulltext/ED324905.pdf.

Cereti, Carlo G. 1996. “Zoroastrian Manuscripts Belonging to the Bhandarkar Institute Collection, Pune.” *East and West* 46, no. 3/4 (December): 441–51. https://www.jstor.org/stable/29757287.

Corff, Oliver. 2016. “The Making of the Pentaglot: Concepts, Data Structures and Tools.” In *Central Asian Sources and Central Asian Research*, edited by Johannes Reckel. Universitätsverlag Göttingen. https://library.oapen.org/bitstream/handle/20.500.12657/31957/1/620950.pdf#page=75.

Daniel, Jennifer. 2022. “The Past and Future of Flag Emoji 🏁” *Did Someone Say Emoji?* (blog), March 28, 2022. https://jenniferdaniel.substack.com/p/the-past-and-future-of-flag-emoji.

Deuchar, Hannah Scott. 2023. “A Case of Multiple Identities: Uncanny Histories of the Arabic Typewriter.” *International Journal of Middle East Studies* 55, no. 2: 238–59. https://doi.org.10.1017/S0020743823000727.

Dickens, Mark. 2020. “Multilingual Christian Manuscripts from Turfan.” *Journal of the Canadian Society for Syriac Studies* 9, no. 1. https://doi.org/10.31826/jcsss-2009-090104.

Dvorak, John. 1994. *Dvorak Predicts: An Insider’s Look at the Computer Industry*. Berkeley, CA: Osborne McGraw-Hill.

Everson, Michael, Karljürgen Feuerherm, and Steve Tinney. 2004. “Final Proposal to Encode the Cuneiform Script in the SMP of the UCS.” ISO/IEC JTC1/SC2/WG2 N2786. June 8, 2004. https://www.unicode.org/L2/L2004/04189-n2786-cuneiform.pdf.

Haralambous, Yannis. 2007. *Fonts & Encodings*. Translated by P. Scott Horne. Sebastopol, CA: O’Reilly. https://hal.science/hal-02112942.

Hellingman, Jeroen. 1997. “Indian Scripts and Unicode.” Posted to the Unicode mailing list, September 3, 1997. https://unicode.org/mail-arch/unicode-ml/Archives-Old/UML009/0597.html.

Hensch, Kurt. 2005. “IBM History of Far Eastern Languages in Computing, Part 1: Requirements and Initial Phonetic Product Solutions in the 1960s.” *IEEE Annals of the History of Computing* 27, no. 1: 17–26. https://doi.org/10.1109/MAHC.2005.9.

Honkapohja, Alpo. 2011. “Multilingualism in Trinity College Cambridge Manuscript O.1.77.1.” In *Foreign Influences on Medieval English*, edited by Jacek Fisiak and Magdalena Bator, 25–45. Frankfurt am Main: Peter Lang. https://doi.org/10.5167/uzh-54984.

Hossain, Anushah. 2024. “Text Standards for the ‘Rest of World’: The Making of the Unicode Standard and the OpenType Format.” *IEEE Annals of the History of Computing* 46 (January–March): 20–33. https://doi.org/10.1109/MAHC.2024.3351948.

Hotchkiss, Griffin. 2016. “Battle of the Fonts.” *Frontier Myanmar*, March 23, 2016. https://frontiermyanmar.net/en/features/battle-of-the-fonts.

INTEXT. 1988. “A Stroke of Multilingual Genius!” Advertisement. *SESAME Bulletin: Language Automation Worldwide*, vol. 2, part 1. Stanford Special Collections M2864 Unicode, box 72, JDB, Printed Material, 1987–1991, folder 3.

Judy, Richard W., and Jane M. Lommel. 1986. “The New Soviet Computer Literacy Campaign.” *Educational Communication and Technology* 34, no. 2 (Summer): 108–23. https://www.jstor.org/stable/30219882.

Kenney, Charles. 1992. *Riding the Runaway Horse: The Rise and Decline of Wang Laboratories*. Boston: Little, Brown, 1992.

Kern, Chris. 2007. “Weird Processing: The Collision of Computers and Cultures at the Voice of America.” *Chris Kern’s Eponymous Website*, October 2007. https://www.chriskern.net/history/computersAndCulturesAtVoa.html.

Kissane, Erin. 2023. "Meta in Myanmar." October 16, 2023. https://erinkissane.com/meta-in-myanmar-full-series.

Komisarenko, S. 1991. Letter to American National Standards Institute. Stanford Special Collections M2864 Unicode, box 61, Cyrillic 1981–1993.

———. 1993. Letter to Kenneth V. Whistler. Stanford Special Collections M2864 Unicode, box 61, Cyrillic 1981–1993.

LaGrow, Nick, and Miri Pruzan. 2019. "Integrating Autoconversion: Facebook's Path from Zawgyi to Unicode." *Engineering at Meta* (blog), September 26, 2019. https://engineering.fb.com/2019/09/26/android/unicode-font-converter/.

Lee, Alx. 2023. "Navajo Prep Students to Publish Second Bilingual Zine." *The Journal*, December 8, 2023. https://www.the-journal.com/articles/navajo-prep-students-to-publish-second-bilingual-zine/.

Lewis, Jason E., and Bruno Nadeau. 2009. "Writing with Complex Type." *Digital Arts and Culture*, December 12–15, 2009. https://escholarship.org/uc/item/37n6g0ww.

Lindia, Matthew S. 2022. "Following Orders: A History of Amharic Typing." *Book History* 25, no. 2 (Fall): 425–42. https://doi.org/10.1353/bh.2022.0015.

LocoScript. 1988. "LocoScript 2 for the Amstrad PCW8256/8512." Advertisement. *SESAME Bulletin: Language Automation Worldwide*, vol. 2, part 1. Stanford Special Collections M2864 Unicode, box 72, JDB, Printed Material, 1987–1991, folder 3.

Miškinienė, Galina. 2009. *Ivano Laucevičiaus kitabas. Lietuvos totorių kultūros paminklas*. Vilnius: Lietuvių kalbos institutas.

Monotype. "Gift of Tongues." Advertisement. Stanford Special Collections M2864 Unicode, box 63, JDB, Indic Standards, 1983–1994, folder 2.

Mullaney, Thomas S. 2017. *The Chinese Typewriter: A History*. Cambridge, MA: MIT Press.

Nemeth, Titus. 2017. *Arabic Type-Making in the Machine Age: The Influence of Technology on the Form of Arabic Type, 1908–1993*. Leiden: Brill, 2017.

Papazian, Hrant. 2023. "Planting Seeds of Diversity." Talk at Face/Interface, Stanford, CA, December 1–2, 2023.

Park, Jae-Hong. 2023. "Mobile Text Readability Improvement Study of Korean Font—Focusing on Google Noto Sans Typeface." *Journal of the Korea Society of Computer and Information* 28, no. 8 (August): 77–86. https://doi.org/10.9708/jksci.2023.28.08.077.

Perkins, C. Ryan. 2017. "Why Did the Muslim World Adopt Print So Late? The Challenges of Perso-Arabic Types and the Lithographic Revolution." Talk at Face/Interface, Stanford, CA, December 2, 2017.

Pratley, Chris. 1999. "Microsoft Office 2000 and Unicode." International Unicode Conference #4, Boston, MA. Stanford Special Collections M2864 Unicode, box 81, folder 2.

Qalam. 1988. "Qalam—a Text System for South Asian Scripts." Advertisement. *SESAME Bulletin: Language Automation Worldwide*, vol. 2, part 1. Stanford Special Collections M2864 Unicode, box 72, JDB, Printed Material, 1987–1991, folder 3.

Rockwell, John, Jr. 1986. "Carving Tibetan in Silicon: A Tibetan Font for the Macintosh." *Newsletter for Asian and Middle Eastern Languages on Computer* 1, no. 3. https://eric.ed.gov/?q=ED312898&id=ED312898.

Scibilia, Ron. 1989. "Technowriters." *Popular Mechanics*, June 1989. https://books.google.com/books?id=BeQDAAAAMBAJ&pg=PA71#v=onepage&q&f=false.

Searle, Steven J. 2004. "Unicode Revisited." *TRON Web*. http://tronweb.super-nova.co.jp/unicoderevisited.html.

Sinha, R. Mahesh K. 2009. "A Journey from Indian Scripts Processing to Indian Language Processing." *IEEE Annals of the History of Computing* 31, no. 1: 8–31. https://doi.org/10.1109/MAHC.2009.1.

Smith, William D. 1971. "Lag Persists for Business Equipment." *New York Times*, October 26, 1971.

Stanton, Andrea. 2021. “Broken Is Word.” In *Your Computer Is on Fire*, edited by Thomas S. Mullaney, Benjamin Peters, Mar Hicks, and Kavita Philip, 213–30. Cambridge, MA: MIT Press.

Syauqi, Ariq. 2023. “Challenges and Opportunities of Adapting Minority Scripts into Digital Platform.” Talk at Face/Interface, Stanford, CA, December 1–2, 2023.

Twardoch, Adam. 2021. Comment on Noto font issue. GitHub, April 20, 2021. https://github.com/notofonts/noto-fonts/issues/2071#issuecomment-823370205.

Tyler, Jane. 1988. “Conference Report: Multilingual Word Processing in Voluntary Organizations, 20 June 1988.” In *SESAME Bulletin: Language Automation Worldwide*, vol. 2, part 1. Stanford Special Collections M2864 Unicode, box 72, JDB, Printed Material, 1987–1991, folder 3.

Xerox. 1985. “VOA Languages: Schedule for Installation (Draft 10/1/85).” Stanford Special Collections M2864 Unicode, box 62, JDB, VOA, 1984–1988, folder 1.

———. n.d. “The Xerox 8010 Speaks Your Language.” Brochures. https://www.digibarn.com/friends/curbow/star/3/index.html;https://www.digibarn.com/friends/curbow/star/2/index.html.