Mining Naturally Romanized Seed Corpora without Romanizations

Adrian Benton[†] **Alexander Gutkin**[◦] **Christo Kirov**[†] **Brian Roark**[‡]
Google Research, [†]New York; [◦]London, UK; [‡]Portland, OR
{adbenton,agutkin,ckirov,roark}@google.com

Abstract

While the Latin script is used informally by speakers of many languages with different native scripts, high quality Latin script corpora for such languages that reflect actual natural romanizations are scarce and often difficult to collect. In this work, we propose a method for mining romanized language corpora in languages for which we do not have any pre-existing samples of naturally romanized text, focusing on Tigrinya as a test case. First we examine the efficacy of learning romanizations for a language based on observed romanizations in other languages that use the same native script. We then extrinsically assess such methods by using a romanization model trained on Amharic data to bootstrap coverage of romanized Tigrinya in a language identification system. Manual evaluation by two L1 and one L2 Tigrinya speakers suggests our method extracts romanized Tigrinya text with acceptably high precision.

Keywords: language identification, informal romanization, low-resource, Ge'ez script, Tigrinya

1. Introduction

Natural language models are trained on text in the languages being modeled. Natural text is a resource that is more plentiful in some languages than in others, as documented in, for example, Simons et al. (2022). Even in languages for which training text is available, not every common mode of language use may be covered by available text, resulting in a continued lack of coverage (see, e.g., Joshi et al., 2019). For example, some languages are written natively in more than one script. Punjabi is written (largely in Pakistan) in Shahmukhi, a Perso-Arabic script, and (largely in India) in Gurmukhi, a Brahmic script (Murphy, 2018). This can and does lead to asymmetries in what kinds of natural language systems are available to Punjabi speakers, depending on the script they use to write it. Another very common (and commonly under-supported) use scenario in many languages is informal romanization, wherein speakers of the language use the Latin script to write the language instead of the native script, even though there is no orthography for the language in the Latin script (Brandt, 2020).

Finding text samples for training natural language models typically involves use of automatic language identification (LID) systems, which themselves depend on collections of text samples in the languages of interest for training. For many even relatively under-resourced languages, open resources such as Wikipedia can provide a starting point for training LID systems. For informally romanized text, however, such datasets are not typically available. To cover these romanized text scenarios in LID systems in the absence of pre-existing text corpora, training data is typically synthesized (Madhani et al. 2023a; Benton et al. 2025) by automatically transliterating text from the native script of the language to the Latin script (automatic romanization).

Due to the large number of regional languages and diversity of scripts, South Asian languages have been the predominant focus of recent work on informally romanized text. Many South Asian languages have publicly available romanization lexicons (Roark et al., 2020; Madhani et al., 2023b), which can be used to train transliteration models for synthesizing romanized text from native script text (Madhani et al. 2023a; Kirov et al. 2024; Benton et al. 2025). Such approaches, thus, assume some initial romanization resources sufficient to enable data synthesis.

What about languages for which such romanization resources are not available? Building a new resource requires native speakers of the language to transliterate a corpus of words or sentences written in one script to their counterparts in another script. The corpus may originally be in the native script of the language, in which case the native speakers must romanize the text to produce the parallel resource; or the corpus may originally be in the Latin script, in which case the native speakers must transliterate the text to the native script.

The quality of the resulting resource will depend both on the quality of the native speaker transliterations, as well as the quality of the input corpus. There are pros and cons to using either originally native or originally Latin script texts. As stated earlier, native script corpora are generally more common. However, elicited romanizations of native script text may lack the naturalness of spontaneously romanized text, which is often produced under informal conditions in many languages. While text originally in the Latin script may provide coverage of common informal registers,

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 2996–3012
11–16 May 2026.

transliteration of this to the native script may be a more challenging task than romanization, particularly if the words or sentences are presented in isolation, outside of the broader context within which they occurred. Resources that include transliterations derived from both native and Latin script texts are likely to provide better coverage of the correspondences between the scripts.

In this paper we examine the use of existing romanization data/models covering other languages that use the same native script as the means for mining initial romanized seed corpora in a language. These seed corpora can be used to elicit romanization lexicons or as further input to improve LID systems.

Before addressing cases lacking any parallel romanization data, we first perform some preliminary controlled experiments looking at multiple South Asian languages using the Devanagari and Bengali scripts, for which we have parallel romanization lexicons. These lexicons allow us to intrinsically assess the quality of romanizations derived from models trained on related languages versus those derived from models trained on the same language. During these preliminary experiments, we present an alternative romanization approach that better generalizes romanization performance across languages.

We then provide an extrinsic evaluation of our approach via language identification of romanized Tigrinya, which, like several neighboring languages spoken in the Horn of Africa, is natively written in the Ge'ez script and is commonly informally romanized (Yaqob, 1997; Bernal, 2005; Conrad, 2010; Afeworki, 2018). Tigrinya has a small Wikipedia but lacks an available romanization lexicon to train a transliteration model for synthesizing text for LID training, hence we rely on parallel Amharic romanizations included in the XTREME-UP benchmark (Ruder et al., 2023) for automatic romanization from the Ge'ez script. The resulting synthesized dataset is added to a language identification model which is used to label Common Crawl text extracted from the MADLAD-400 dataset (Kudugunta et al., 2023). Samples of text labeled as romanized Tigrinya are then assessed to determine whether they are actually romanized Tigrinya. Results under various conditions show precision of 90% or higher, with general language-agnostic filtering yielding very high precision at the expense of some recall.

2. Background

Romanization Spontaneous, informal use of the Latin script can occur in languages that use other scripts for many reasons, such as lack of (or unfamiliarity with) available native script keyboards. Although formal romanization systems are widely used for text entry in some languages,[1] for many languages and cultures, adoption of formal systems is modest, so that romanizations can take many forms. The lack of orthography in the Latin script and corresponding spelling variability can make processing – or even identifying – Latin script text challenging. Romanization lexicons can help model these informal romanizations.

Romanization lexicons, e.g., the Dakshina (Roark et al., 2020) and Aksharantar (Madhani et al., 2023b) datasets, provide one or more possible romanizations for each given native script word. For example, the Aksharantar romanization lexicon for Hindi contains four distinct entries for the word "सवेरा" (meaning *dawn*): *savera*, *saveraa*, *sawera*, and *saweraa*. Key distinctions here include whether the final long vowel is represented with a single or double Latin script vowel, and whether the second syllable starts with a consonant best represented by 'v' or 'w'. Given a parallel resource of this sort, a sequence-to-sequence model can be trained to convert strings in the native script to strings in the Latin script. This sort of romanized text synthesis has been used to dramatically improve language identification of informally romanized text (Benton et al., 2025).

Absent such language-specific resources, one might try using a universal romanization system, which is typically defined for the script, to provide a canonical Latin script spelling for each word. This canonical spelling does not necessarily correspond to what speakers of the language are likely to produce spontaneously, and lacks the variability of naturally occurring romanizations. Nevertheless, a universal romanization system may be useful absent any alternatives. Both the `uroman` (Hermjakob et al., 2018) and `uconv`[2] systems provide such universal romanizations.

With respect to the Ge'ez script, general Ethiopic romanization systems have been proposed, including Firdyiwek and Yaqob (1997), which is used within the HornMorpho morphological system (Gasser, 2011) for three regional languages, including Tigrinya. In practice, Tigrinya speakers do not follow this formal ASCII romanization scheme. The use of romanization can vary widely, even between language communities. For example, Fallon (2006, 2008) discusses Blin, a vulnerable Cushitic language of Eritrea that exists in the state of *digraphia*. Despite an official Latin orthography introduced for the language by the Eritrean government, the Ge'ez script is preferred within the diaspora as the means of better integrating into Tigrinya speaking communities abroad.

[1] Such as *pinyin* in Chinese.

[2] `uconv` is part of the ICU libraries provided by the Unicode Consortium: `https://icu.unicode.org`.

Mining Romanized Text Previous work on mining informally romanized corpora has largely focused on languages or dialectal variants without established orthographies. The need for such corpora is acute (Blaschke et al., 2023; Ramponi, 2024). Millour and Fort (2020) provide many examples of the use of informal orthographic registers in "newly written languages" from across the world. Beyond manual language preservation or crowd-sourcing based approaches to corpus collection, such as the corpus of Gronings, a Low Saxon dialect (Sekeres et al., 2024), prior work has mined under-resourced Latin text using automatic pipelines consisting of a series of filtering steps. Examples include the mining pipeline for Swiss German (Linder et al., 2020) and a system for collecting romanized Hindi lyrics (Gupta et al., 2012). Relying on lists of high precision allow/block lists has been used to identify various French creoles (Dent et al., 2025). The method we propose here is complementary to word list filtering approaches.

We are unaware of existing work focused on mining informally romanized text for native Ge'ez script languages, including Tigrinya. The XTREME-UP benchmark (Ruder et al., 2023) includes informally romanized Amharic text, which was manually elicited from native speakers by requesting romanization of Ge'ez script Wikipedia text. The GeezSwitch (Gaim et al., 2022) language identification system solely applies to text written in the Ge'ez script. Recent Ethiopia-centric large language models (LLMs) and benchmarks such as EthioLLM (Tonja et al., 2024) also do not appear to include informally romanized training text,[3] nor do recent large-scale, broad-coverage harvested datasets, such as GlotCC (Kargaran et al., 2024), FineWeb2 (Penedo et al., 2025) or DCAD-2000 (Shen et al., 2025).

Before moving on to examine romanized Tigrinya, we first present the romanization methods we use throughout the paper, along with some preliminary experiments to intrinsically assess the usefulness of romanization models trained on related languages.

3. Romanization Methods

In this paper, our experiments require romanization from three native scripts: Devanagari, Bengali and Ge'ez. In this section, we present the methods we use for the specific languages and scripts included in the experiments.

[3]The three Latin script languages included as pretraining data in EthioLLM are English, Oromo, and Somali, all of which are formally written in the Latin script.

3.1. Universal Romanizers

As baseline romanizers, as well as for the preprocessing discussed in Section 3.2.2 below, we use the two universal romanization methods mentioned in Section 2: `uroman` and `uconv`. The `uroman` system produces undiacritized Latin script output for all of the scripts, with certain distinctions such as vowel length or gemination indicated by doubling. In contrast, most romanization systems[4] provided by `uconv` use diacritics to make such distinctions. For example, our earlier example Hindi word "सवेरा", which had four different romanizations in the Aksharantar lexicon, is romanized by `uroman` as "saveraa" and by `uconv` as "savērā". Stripping all diacritics from the base characters (e.g., converting savērā to savera) results in something closer to naturally occurring romanizations, though such a conversion leaves phenomena such as schwa deletion unaddressed. We will refer to `uconv` romanizations stripped of diacritics as `uconv`′. Note that both the `uroman` and `uconv`′ romanizations (*saveraa* and *savera*) are in the set of this word's romanizations in the Aksharantar lexicon, alongside two others.

3.2. Romanization Data

The transliteration models that we use for romanization in this paper (see Section 3.3) are trained from parallel romanization lexicons. These romanization lexicons are sourced differently for the Brahmic scripts (Bengali and Devanagari) from Ge'ez.

3.2.1. Brahmic Scripts

Brahmic script romanization lexicons for this study come from the Aksharantar dataset[5] (Madhani et al., 2023b), which contains romanization lexicons for 20 of the 22 scheduled languages of India, including 7 languages natively using the Devanagari script and 2 natively using the Bengali script. Each language's lexicon is divided into training, validation and testing partitions, and here we evaluate solely on validation partitions after training romanization models on training partitions.

The seven languages (ISO 639-3 codes shown in parantheses) that use the Devanagari script mostly come from the Indo-Aryan language family: the modern languages Hindi (`hin`), Konkani (`kok`), Maithili (`mai`), Marathi (`mar`) and Nepali (`nep`); and

[4]For two of the scripts we investigate in this paper, Devanagari and Bengali, `uconv` provides a single romanization system. For the Ge'ez script, however, there are multiple supported romanization systems available in `uconv`, and for this work we use the Ethiopic-Latin/BGN transliteration.

[5]https://huggingface.co/datasets/ai4bharat/Aksharantar

the historic language Sanskrit (san). The Tibeto-Burman language Bodo (brx) also uses Devanagari. Two Indo-Aryan languages use the Bengali script in their native writing systems: Assamese (asm) and Bangla (ben). The lexicon sizes vary—Bodo is the smallest, with under 34k unique native script words in the training partition.

3.2.2. Intermediate Universal Romanization

In addition to using universal romanization methods as baselines for the Brahmic script romanization experiments in Section 4, we also assess their use as intermediate representations for data-driven romanization. In such an approach, the native script string is first passed through a universal romanization system (in our experiments uconv), which is then transliterated using models trained on romanization lexicons, with the universal romanization produced by uconv on the source side.

The rationale for using such a Latin script intermediate representation is that different languages can and do use different character subsets of the scripts that they share, impacting coverage. The Brahmic and Ge'ez scripts are *abugidas*, where a character generally represents a consonant-vowel sequence—often referred to as *orthographic syllables* or *aksara* in the case of Brahmic scripts—and conversion first to an alphabetic representation, where vowels and consonants are separate characters, may allow for easier generalization. This is particularly acute for Ge'ez, where each grapheme is represented by a single unicode codepoint. In contrast, the Brahmic scripts use a range of vowel diacritics, so that even if a particular consonant/vowel combination is not observed in training, some generalization can be drawn based on observation of the diacritic in other contexts. In Ge'ez, each grapheme is atomic, hence such cross-grapheme generalization is challenging. For instance, the orthographic syllable for /mu/ is represented as ሙ (U+1219) in Ge'ez, whereas it is represented as মু (two codepoints: U+092E and U+0941) in Devanagari. After universal romanization, however, the consonant and vowel are represented separately, yielding improved opportunities for generalization.

To operationalize such an approach, one takes an existing romanization lexicon and replaces the native script word with the uconv romanized word. This modified lexicon can then be used to train a transliteration model using whatever method is used with the original dataset. Then, at time of inference, the input string is romanized using the universal romanization system prior to being transliterated using the model.

3.2.3. Ge'ez Script

For the lone Tigrinya Ge'ez script condition, we do not have a pre-existing romanization lexicon at our disposal. Rather, we construct a romanization lexicon for Amharic from the full sentence manually romanized Amharic data provided in the XTREME-UP benchmark (Ruder et al., 2023).[6] This consists of 2,120 parallel sentences in aggregate.

We constructed a romanization lexicon from this by performing a (whitespace-delimited) word alignment between words in the original native script strings with their romanized counterparts. This word alignment involved several steps of preprocessing. First, whitespace normalization was performed, since the Ethiopic Wordspace character (U+1361, ፡) was used inconsistently in the data, even on the Latin script side of the parallel sentences. All whitespace, including Ethiopic Wordspace, was replaced with a single ASCII space character. Ethiopic number representation differs from common decimal digit notation (Chrisomalis, 2010; Meyer and Wakjira, 2023) and direct uconv romanization fails to romanize these correctly. For example, the string "፫፻፹፯" was romanized as "3100807" (independently romanizing each of the characters) instead of the correct numerical value "387". Further, both Ethiopic and decimal digit representations were used inconsistently on the Ge'ez script side of the parallel data, so a custom conversion to decimal values was applied to all Ethiopic number strings.

Passing the Ge'ez side of the parallel sentences through uconv yielded an additional intermediate universal romanized version of each example. For ease of word alignment, we discarded sentences where the number of whitespace-delimited words in the uconv romanized string differed from the number of words in the human produced romanizations. This resulted in 440 sentences being discarded, i.e., yielding 1,680 sentences with a one-to-one word mapping as provided in a romanization lexicon. From this resource, models can be built to romanize either directly from the Ge'ez script or using the uconv intermediate representation, as discussed in Section 3.2.2.

3.3. Transliteration Modeling

In this work, we follow Benton et al. (2025) in using pair n-gram models (Bisani and Ney, 2008) in romanization systems for synthesizing training data for language identification (LID) of romanized text. As with that work, we use methods described in

[6] Available by following the download instructions at https://github.com/google-research/xtreme-up. The training, validation and test splits are from Ethi2Latn.am.jsonl files under transliteration subdirectory.

Kirov et al. (2024) for training pair n-gram models from lexicons of paired source/target words. Kirov et al. (2024) showed that such models are competitive even with vastly more expressive pretrained neural models for romanization, and Benton et al. (2025) showed that sampling from such models yields LID improvements, likely due to the spelling variability resulting from a lack of orthography.

We infer character alignments between source and target words in the lexicon via expectation maximization (Dempster et al., 1977). Words are constrained to align monotonically, and characters on either side are optionally allowed to align to the empty string ϵ (e.g., in the case that one word contains more unicode codepoints than the other). For example, in the XTREME-UP transliteration training set, the Amharic word "ማዕዘን" is realized as "maizen" in the Latin script. A possible alignment between these two strings would be, for example:

ማ:m ϵ:a ዕ:i ዘ:z ϵ:e ን:n.

Since Ge'ez character codepoints generally represent orthographic syllables, they may be implicitly associated with more than one Latin script character (a consonant followed by a vowel). This will require use of ϵ in the alignment, as with ማ in this example. Exceptions in the above example are "ዕ", corresponding to IPA /ʕ/, which may not be represented explicitly in a natural romanization (sometimes indicated by an apostrophe), and "ን" which is in a form that suppresses the inherent vowel. If using uconv output (māʿəzan) as an intermediate representation, a possible alignment would be:

m:m ā:a ʿ:ϵ ə:i z:z a:e n:n

Models can be trained from either sort of alignment, allowing a straightforward exploration of the utility of such an intermediate representation.

After alignment, each source/target character pair is treated as a single token, each string of pair tokens is treated as a separate sequence, and an n-gram language model is fit over this corpus of pair token strings. We represent the resulting pair n-gram model as a weighted finite state transducer (WFST), by treating each pair's source-side character (including ϵ) as input, and each target-side character (including ϵ) as output. This transducer is our learned transliteration model, mapping from source to target words. At inference time, we can decode transliterations based on the likelihood of producing the output conditioned on the input, under the weights of the pair n-gram language model. Shortest-path algorithms—including for extraction of k-best paths—are efficient, and for inference we make use of the OpenFst library (Allauzen et al., 2007). See Kirov et al. (2024) for further details and links to code for training and inference.

4. Romanization Experiments

For Aksharantar validation sets of languages using Bengali and Devanagari scripts, we assess intrinsic performance of various romanization systems, including baseline universal romanization systems and those based on data-driven romanization, trained on different languages. Among other things, we find that (1) trained data-driven systems outperform baseline universal romanization systems; and (2) a two-stage approach, which first performs universal romanization followed by data-driven transliteration, achieves the best cross-lingual transfer. Extrinsic validation is in Section 6.

4.1. Evaluation

We noted above that the quantity of training data available in Aksharantar differs between languages, with all languages having at least 33k unique native script words in their lexicons. To improve comparability across conditions, for each language, we independently train 10 romanization models as follows: 30k words are randomly selected from the language's Aksharantar training lexicon, and all romanizations for the selected words are included in the training data used to train a pair n-gram romanization model as described in Section 3.3. Then, for each condition we run 10 trials using these independently trained models and report the mean performance across trials.[7]

Using these romanization systems, words in the validation partition for each language that shares the same native script are romanized by finding the most likely (shortest path) romanization. We follow Kirov et al. (2024) in evaluating romanization using minimum character error-rate percentage over the set of reference romanizations included in the validation partition for that word. For each reference romanization, character error rate percentage (CER%) is calculated for the system romanization, i.e., the Levenshtein distance per 100 reference characters. The smallest CER% over the reference set is the minimum CER% (mCER%).

4.2. Results

Table 1 presents mCER% for a range of conditions. The upper Table 1(a) presents methods taking the native script word directly into the romanization system. For each row, representing a target language for romanization, we show results for romanization systems trained on languages using the same native script, along with two universal romanization baselines. Unsurprisingly, the error rates along the diagonal (trained on the target language) are the best for each language (hence

[7]Variances are reported in Appendix Section C.

(a) native script input:

Lang	Script	romanization system										
		brx	hin	kok	mai	mar	nep	san	asm	ben	uconv′	uroman
brx	Deva	**4.8**	31.1	38.7	32.1	31.6	33.2	53.6	N/A		42.9	54.4
hin		29.6	**8.1**	15.1	11.6	11.5	12.0	27.4			24.5	31.1
kok		32.0	13.3	**11.0**	15.4	11.7	17.0	18.1			20.6	23.9
mai		29.4	9.9	12.6	**7.1**	11.5	11.2	16.4			20.9	21.3
mar		25.0	5.6	7.3	8.9	**3.8**	9.4	23.8			14.8	30.4
nep		26.2	8.8	12.8	10.8	9.4	**3.9**	23.5			15.1	26.0
san		37.2	15.3	11.1	13.9	14.0	15.5	**3.5**			13.6	10.2
asm	Beng	N/A							**4.7**	30.7	39.3	48.9
ben									21.1	**7.8**	22.6	32.7

(b) uconv **intermediate:**

Lang	Script	romanization system								
		brx	hin	kok	mai	mar	nep	san	asm	ben
brx	Deva	**5.2**	31.1	38.5	32.6	31.5	33.7	53.7	N/A	
hin		28.3	**7.9**	13.4	11.5	9.9	11.6	27.4		
kok		30.5	12.8	**10.7**	14.5	11.6	15.9	17.8		
mai		28.9	9.3	11.5	**7.0**	9.9	10.8	16.7		
mar		23.4	5.1	7.2	7.8	**3.6**	8.0	23.5		
nep		25.4	8.1	11.7	10.8	8.5	**3.9**	23.6		
san		37.3	13.6	9.5	13.0	13.1	15.2	**3.6**		
asm	Beng	N/A							**5.4**	24.6
ben									19.4	**7.4**

Table 1: Minimum character error-rate percentage (mCER%) for romanization of languages written in either Devanagari or Bengali using systems trained on romanization lexicons in different languages. Table (a) contains results when romanizing directly from native script input, and includes two general romanization system baselines: uconv′ is the output of the ICU uconv utility, stripped of any diacritics; and uroman universal romanization system (Hermjakob et al., 2018). Table (b) presents romanization results when the native script input is first romanized via uconv, then further transliterated via models trained on romanization lexicons of various languages.

bolded). For the modern Indo-Aryan languages—i.e., excluding Bodo (brx) and Sanskrit (san)—training on another modern Indo-Aryan language always provides substantially lower error rates than either of the universal baselines. From this we can conclude that learning a transliteration model on data, even with a language mismatch between train and inference time, provides utility for generating natural romanizations. This holds across languages written in both Devanagari and Bengali.

Bodo error rate is very degraded when training on other languages. Some of this likely comes from gross linguistic differences, as Bodo belongs to a different language family. However, we also note that certain common vowel romanizations in Bodo are quite different from other languages. For example, for the Sanskrit-origin word "आरोग्य" (meaning *health*), all six of the modern languages using Devanagari have this word in their romanization lexicon, and in Bodo it is romanized as "arwgyo" in contrast to the other languages, which all have "arogya" in their set of romanizations. Bodo was originally written using the Latin script (Sarmah, 2014), and conventionally 'w' is used to indicate a close, back, unrounded vowel. The common use of 'w' for vowel romanizations does not appear in the other languages, hence cannot be learned from data in those other languages.

Sanskrit, interestingly, is best served by the uroman universal romanization system, which indicates that Sanskrit romanizations are more formal in representation of, say, vowel length and gemination. Since it is primarily a liturgical language, this increased formality is unsurprising.

The lower Table 1(b) presents error rates when using uconv output as an intermediate representation. This does not change performance much when training on the target language itself, but provides modest improvements to most other conditions. For the Bengali script, Assamese shows large error rate reductions with this method. This is largely due to differences between Bengali and Assamese in their use of various 'Ra' letters in the Bengali script. A complete lack of coverage in the Bengali romanization lexicon of two 'Ra' variants used heavily in Assamese (and also romanized by uconv as 'ra') was fixed by the intermediate universal romanization. Beyond improvement in the face of such gross mismatch, however, this technique mostly helps and, at least for related modern languages, never hurts transfer.

These results demonstrate that being within the same language family does improve transfer versus across language families, but that is no guarantee of good performance. Assamese, for example, even with the uconv intermediate representa-

tion, has a mCER% around 25. Still, it seems clear that using such an intermediate representation is helpful for improving transfer.

5. Romanized Text Mining Methods

We now describe our methods for extracting romanized text for under-resourced languages from a generic text corpus (Figure 1). We begin by describing each of the ingredients necessary to execute the recipe, both models and datasets. We follow by practically describing each of these ingredients for each of the languages in this work.

Necessary Ingredients Let TL be the target language for collecting naturally romanized text, and let HRL be a higher resource (linguistically related) language written in the same native script. We assume access to the following resources:

- An *HRL romanization lexicon*: a list of words written in the native script, paired with the same word written in Latin script, as described in Section 3.2. The romanizations should be representative of natural romanizations produced by native speakers.
- A universal *TL script romanization system*, as described in Section 3.1. This is to permit an intermediate representation closer to the desired natural romanization that the native script, which was shown to be beneficial in Section 4.
- *TL native script sample* sentences. In our experiments, these are on the order of 1000s of Wikipedia sentences. At the time of writing, Wikipedia covers 342 languages (of which Tigrinya is ranked number 337 based on article count).
- A wide coverage *LID training set* that does not already cover TL in the Latin script. Several publicly available datasets can be used to this end, including OpenLID (Burchell et al., 2023), GlotCC (Kargaran et al., 2024), and DCAD-2000 (Shen et al., 2025), although the latter two are the result of applying an existing LID model to web documents.
- Finally, we assume access to a general *web text corpus*. Various corpora are publicly available, and are typically derived from the Common Crawl repository,[8] e.g., MADLAD-400 (Kudugunta et al., 2023) or multilingual C4 (Xue et al., 2021). This constitutes the set to be mined, and documents need not be filtered by an existing LID model.

Our method is sketched in Figure 1.

Preprocessing Details Preprocessing of the romanization lexicons and Wikipedia native script Tigrinya examples was performed as described in Section 3.2.3.

The web text upon which we will attempt to extract romanized Tigrinya samples is drawn from the noisy partition of MADLAD-400, which has not been filtered with language identification. This dataset includes a de-duplicated union of all of the Common Crawl snapshots available in August of 2022. We perform newline segmentation, so our extraction is at the paragraph rather than document level. We designed a system, which we call *Common Scrawl*, consisting of a series of filters that allow us to focus on a script of interest (in this case Latin) and avoid common sources of noise in web text, such as non-linguistic text or boilerplate and other classes of text that are not representative of spontaneous natural language. All paragraphs accessed for these experiments were minimally filtered to consist predominantly of Latin script text.

For the current experiments, it suffices to consider two disjoint paragraph subsets: text which makes it through all of the filtering stages (`clean`), and text which is discarded by some filtering stage (`discarded`). This allows us to compare the precision of extracted text before and after filtering. Full details of the filtering stages are described in Appendix Section A.

Tigrinya In the case of the TL Tigrinya, we consider the HRL to be Amharic. We make use of `uconv` for intermediate universal romanization.

We collected a 5,005 sentence sample of Tigrinya, containing a total of 77,493 tokens from Wikipedia for use as a native script sample.[9] This we then romanized by: (1) extracting the k-best romanizations for each token; (2) softmax normalizing the scores from the transliteration model over the k-best list; and (3) sampling from this list according to the resulting distribution. We made ten passes over the native script corpus, resulting in ten distinctly sampled corpora. These were concatenated to form the synthetic romanized Tigrinya training set for our LID filter.

For the LID training set, we also included 100,000 English and 10,000 sentences from each of the 303 Latin script non-English languages from the MADLAD-400 dataset. We also included 10,000 sentences of synthetically romanized training data for the twenty romanized South Asian languages provided in the Bhasha-Abhijnaanam South Asian LID task (Madhani et al., 2023a), as these languages are frequently written in the Latin script on the web but have poor coverage and qual-

[9] https://ti.wikipedia.org/

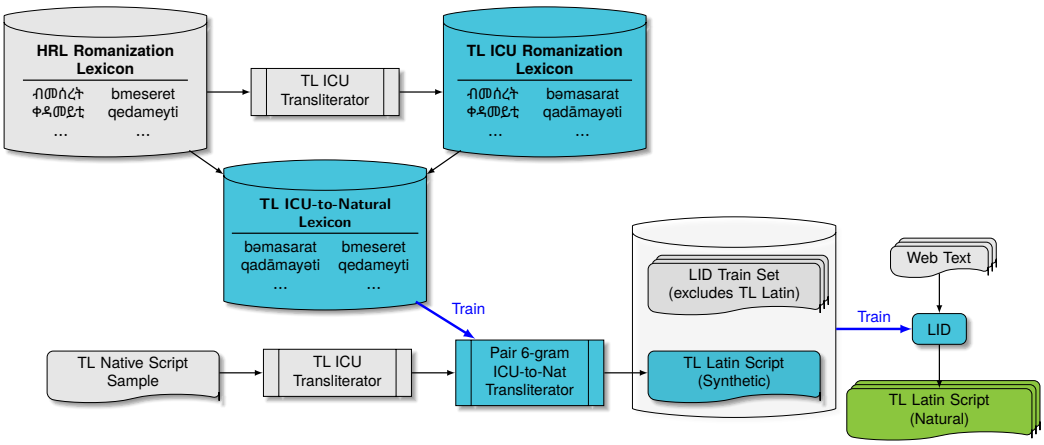

Figure 1: Flow chart of the proposed mining method. Cylinders indicate transliteration lexicons, tape symbols indicate text document(s), and models are indicated by rectangles. Gray indicates a preexisting resource, blue indicates derived, and green indicates the resulting product: naturally romanized text from the target language.

ity issues in MADLAD.[10] We train a fastText LID filter (Joulin et al., 2016) on this set, and apply it to a cleaned subset of the MADLAD noisy partition to identify Tigrinya sentences with at least 50% confidence, and finally pass only the most confident sentences for annotators to rate.

Language Identification Model In this work, we train a LID model to filter naturally romanized Tigrinya. While large language models can also perform admirably as LID models, they are expensive to run in practice, and lower capacity models have been shown to perform well as web-scale filters (Kargaran et al., 2023; Benton et al., 2025). We strip punctuation from any text before training/inference and set fastText LID hyperparameters to those used in Benton et al. (2025).

Post-filtering Details After LID filtering, we applied some light cleaning to spare annotators from rating noisy text, or text that was clearly not natural language. Sampled sentences had to contain at least three words observed in the synthetic Tigrinya training set, have at least 70% ASCII characters, and no more than 10% of the space-delimited tokens could be hashtags or usernames. In addition, we removed any sentence with low character trigram entropy relative to the set of sentences identified as Latin script Tigrinya (in the second decile or less). This was to avoid unnaturally repetitive texts that may have passed the LID filter. We also filtered out any string that contained a spam-indicative word, e.g., *mp3*, *download*.

[10] For instance, Kudugunta et al. (2023) highlights several quality issues for these languages in Appendix 4, "Monolingual Data Details".

Corpus	N	Majority	Unanimous
`all`	100	95	93
`clean`	100	99	96
`discarded`	100	84	84

Table 2: Number of sentences agreed to be romanized Tigrinya by human annotators, either by majority vote or unanimously. Note that the `discarded` and `clean` corpora are proper subsets of `all`.

Tigrinya Manual Evaluation We worked with three annotators who were familiar with Tigrinya. Two were native speakers, and the third was a native speaker of a related language and fluent enough to be able to identify whether a piece of text was actually romanized Tigrinya.

We mined the 100 most confident Tigrinya paragraphs from (a) the entire preprocessed Web Text corpus (`all`); (b) the filtered corpus (*clean*); and (c) what was discarded by filtering (`discarded`), which is the complement of the `clean` subset. We took the union of these newline-delimited paragraphs, resulting in 204 paragraphs total, and provided these to annotators, who coded whether each sentence was actually romanized Tigrinya. We considered each of these sets as a base web text corpus in order to additionally extrinsically validate for our web text filtering pipeline, offering a rough estimate of recall lost due to filtering.

Annotator instructions and a selection of annotated sentences and comments are presented in Section B.

6. Tigrinya Mining Evaluation

Table 2 breaks down the proportion of genuine romanized Tigrinya sentences examples by source

corpus and annotator reconciliation method. Of the 204 most confident Tigrinya sentences mined, annotators unanimously agreed that 183 (89.7%) were Tigrinya, with majority agreement on 186 (91.2%) sentences; 17 (8.3%) were unanimously agreed to not be Tigrinya. The remaining four sentences with annotator disagreement were typically sentences mixing words from different languages. For example, the use of "habibi", a borrowing from Arabic, in "Korchach nay benay eka habibi", or "Yew tse' ana eyu kabzi'atom" which one annotator deemed to mostly contain Amharic even though they agreed that "kabzi'atom" was Tigrinya.

Although our initial filtering throws away text containing legitimate romanized Tigrinya sentences (as evidenced by 84% unanimous agreement on the `discarded` set), this precision may not be acceptable for subsequent consumption. The benefit of filtering is evidenced by the precision on the `clean` set. There, 99 out of 100 sentences were deemed Tigrinya by majority vote, with unanimous vote disgreement driven by possible borrowings and code mixing. Many of the false positives on the discarded corpus are egregiously unnatural, including strings of URLs ("webnet.gov.bz webnet.net.bz webnet.org.bz webnet.bj webnet.gouv.bj webnet.mil.bj") and nonsensical cycling through character n-grams ("zta-ztz : zte ztg ztl ztm ztn ztp ztq zts ztt ztu ztx zty"). While the recall in our filtering process could be improved, the mining procedure we describe here is independent of the form of the web corpus. Depending on the precision required, one can apply more or less strict quality filtering constraints.

7. Discussion and Conclusions

In this paper, we have demonstrated the utility of synthetic romanizations taken from models trained on related languages for training language identification systems for romanized text. We presented evidence in support of using intermediate universal romanization to improve such cross-lingual transfer. The precise conditions for successful romanization model transfer remain unclear: even related regional languages Assamese and Bangla yielded relatively poor cross-lingual transfer.

8. Acknowledgments

The authors thank Daniel Adonai, Milen Haile, and Helen Mehreteab for their help with experiments in this paper, and Cibu Johny for useful suggestions for an earlier draft of this paper.

9. Ethics Statement

The goal of this work is to improve the field's ability to include balanced and inclusive training sets for natural language models. Including representative text from generally under-represented languages and use-scenarios improves the accessibililty of natural language systems, particularly for important, widely-used and chronically under-represented modes such as informal romanization.

10. Limitations

Given the commonly informal nature of romanization in many languages, some romanization styles may be poorly captured by our methods, leading to a bias in resulting models. Indeed, even when explicit romanization schemes are officially recommended, there may be competing schemes that are variably captured by our methods, hence modeling bias can arise whether romanization is informal or not. The choice of a high resource language upon which to base romanization transfer learning may not be as clear as it was with Amharic for Tigrinya, and this choice is not something we explore in this paper.

11. Bibliographical References

Niat Gebremichael Afeworki. 2018. Eritrean nationalism and the digital diaspora: Expanding diasporic networks via Twitter. Master's thesis, African Studies, University of California (UCLA), Los Angeles, CA.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of 12th International Conference on Implementation and Application of Automata (CIAA)*, pages 11–23, Prague, Czech Republic. Springer.

Adrian Benton, Alexander Gutkin, Christo Kirov, and Brian Roark. 2025. Improving informally romanized language identification. In *Proceedings of EMNLP*, forthcoming.

Victoria Bernal. 2005. Eritrea on-line: Diaspora, cyberspace, and the public sphere. *American Ethnologist*, 32(4):660–675.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

Verena Blaschke, Hinrich Schuetze, and Barbara Plank. 2023. A survey of corpora for Germanic low-resource languages and dialects. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 392–414, Tórshavn, Faroe Islands. University of Tartu Library.

Carmen Brandt. 2020. From a symbol of colonial conquest to the *scripta franca*: The Roman script for South Asian languages. In Carmen Brandt and Hans Harder, editors, *Wege durchs Labyrinth: Festschrift zu Ehren von Rahul Peter Das*, pages 1–36. CrossAsia-eBooks, Heidelberg, Germany.

Ralf Brown. 2014. Non-linear mapping for improved identification of 1300+ languages. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 627–632, Doha, Qatar. Association for Computational Linguistics.

Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023. An open dataset and model for language identification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020. Language ID in the wild: Unexpected challenges on the path to a thousand-language web text corpus. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6588–6608, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Stephen Chrisomalis. 2010. *Numerical notation: A comparative history*, chapter Alphabetic Systems. Cambridge University Press, Cambridge, UK.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Bettina Conrad. 2010. *"We are the Prisoners of our Dreams": Long-distance Nationalism and the Eritrean Diaspora in Germany*. Ph.D. thesis, Department of Social Sciences, University of Hamburg, Hamburg, Germany.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: series B (methodological)*, 39(1):1–22.

Rasul Dent, Pedro Ortiz Suarez, Thibault Clérice, and Benoît Sagot. 2025. Krèyolid from language identification towards language mining. *arXiv preprint arXiv:2503.06547*.

Paul D. Fallon. 2006. Blin orthography: a history and an assessment. In *Selected proceedings of the 36th Annual Conference on African Linguistics (ACAL)*, pages 93–98, Statesboro, GA.

Paul D. Fallon. 2008. Language development in Eritrea: The case of Blin. In Barbara Soukup, Lyn Fogle, Jia Jackie Lou, Kendall A. King, and Natalie Schilling-Estes, editors, *Endangered and Minority Languages and Language Varieties: Sustaining Linguistic Diversity*, chapter 10, pages 145–158. Georgetown University Press, Washington DC.

Yitna Firdyiwek and Daniel Yaqob. 1997. The system for Ethiopic representation in ASCII. https://www.researchgate.net/publication/2682324_The_System_for_Ethiopic_Representation_in_ASCII.

Fitsum Gaim, Wonsuk Yang, and Jong C. Park. 2022. GeezSwitch: Language identification in typologically related low-resourced East African languages. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6578–6584, Marseille, France. European Language Resources Association.

Michael Gasser. 2011. HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya. In *Proceedings of Conference on Human Language Technology for Development*, pages 94–99, Alexandria, Egypt. https://github.com/hltdi/HornMorpho.

Kanika Gupta, Monojit Choudhury, and Kalika Bali. 2012. Mining Hindi-English transliteration pairs from online Hindi lyrics. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2459–2465, Istanbul, Turkey. European Language Resources Association (ELRA).

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Bhakti Jadhav, Himanshu Dutta, Shruti Kanitkar, Malhar Kulkarni, and Pushpak Bhattacharyya. 2025. An introduction to computational identification and classification of Upamā alaṅkāra. In *Computational Sanskrit and Digital Humanities - World Sanskrit Conference 2025*, pages

1–14, Kathmandu, Nepal. Association for Computational Linguistics.

Pratik Joshi, Christain Barnes, Sebastin Santy, Simran Khanuja, Sanket Shah, Anirudh Srinivasan, Satwik Bhattamishra, Sunayana Sitaram, Monojit Choudhury, and Kalika Bali. 2019. Unsung challenges of building and deploying language technologies for low resource language communities. In *Proceedings of the 16th International Conference on Natural Language Processing*, pages 211–219, International Institute of Information Technology, Hyderabad, India. NLP Association of India.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. FastText.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Amir Hossein Kargaran, Ayyoob Imani, François Yvon, and Hinrich Schuetze. 2023. GlotLID: Language identification for low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6155–6218, Singapore. Association for Computational Linguistics.

Amir Hossein Kargaran, François Yvon, and Hinrich Schütze. 2024. GlotCC: An open broad-coverage CommonCrawl corpus and pipeline for minority languages. *Advances in Neural Information Processing Systems*, 37:16983–17005.

Christo Kirov, Cibu Johny, Anna Katanova, Alexander Gutkin, and Brian Roark. 2024. Context-aware transliteration of Romanized South Asian languages. *Computational Linguistics*, 50(2):475–534.

Sneha Kudugunta, Isaac Caswell, Biao Zhang, Xavier Garcia, Derrick Xin, Aditya Kusupati, Romi Stella, Ankur Bapna, and Orhan Firat. 2023. MADLAD-400: A multilingual and document-level large audited dataset. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:67284–67296.

Lucy Linder, Michael Jungo, Jean Hennebert, Claudiu Cristian Musat, and Andreas Fischer. 2020. Automatic creation of text corpora for low-resource languages from the Internet: The case of Swiss German. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2706–2711, Marseille, France. European Language Resources Association.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea. Association for Computational Linguistics.

Yash Madhani, Mitesh M. Khapra, and Anoop Kunchukuttan. 2023a. Bhasa-Abhijnaanam: Native-script and romanized language identification for 22 Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 816–826, Toronto, Canada. Association for Computational Linguistics.

Yash Madhani, Sushane Parthan, Priyanka Bedekar, Gokul Nc, Ruchi Khapra, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Khapra. 2023b. Aksharantar: Open Indic-language transliteration datasets and models for the next billion users. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 40–57, Singapore. Association for Computational Linguistics.

Ronny Meyer and Bedilu Wakjira. 2023. Scripts and writing in Ethiopia. In Ronny Meyer, Bedilu Wakjira, and Zelealem Leyew, editors, *The Oxford Handbook of Ethiopian Languages*, Oxford Handbooks, pages 86–100. Oxford University Press, Oxford, UK.

Alice Millour and Karën Fort. 2020. Text corpora and the challenge of newly written languages. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 111–120, Marseille, France. European Language Resources association.

Anne Murphy. 2018. Writing Punjabi across borders. *South Asian history and culture*, 9(1):68–91.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. 2025. FineWeb2: One pipeline to scale them all — adapting pre-training data processing to every language. *arXiv preprint arXiv:2506.20920*. https://github.com/huggingface/fineweb-2.

Alan Ramponi. 2024. Language varieties of Italy: Technology challenges and opportunities. *Transactions of the Association for Computational Linguistics*, 12:19–38.

Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J. Mielke, Cibu Johny, Isin Demirsahin,

and Keith Hall. 2020. Processing South Asian languages written in the Latin script: the Dakshina dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2413–2423, Marseille, France. European Language Resources Association.

Sebastian Ruder, Jonathan H. Clark, Alexander Gutkin, Mihir Kale, Min Ma, Massimo Nicosia, Shruti Rijhwani, Parker Riley, Jean-Michel A Sarr, Xinyi Wang, John Wieting, Nitish Gupta, Anna Katanova, Christo Kirov, Dana L. Dickinson, Brian Roark, Bidisha Samanta, Connie Tao, David I. Adelani, Vera Axelrod, Isaac Caswell, Colin Cherry, Dan Garrette, Reeve Ingle, Melvin Johnson, Dmitry Panteleev, and Partha Talukdar. 2023. XTREME-UP: A user-centric scarce-data benchmark for under-represented languages. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1856–1884, Singapore. Association for Computational Linguistics.

Satyendra Kumar Sarmah. 2014. Script movement among the Bodo of Assam. *Proceedings of the Indian History Congress*, 75:1335–1340.

Hedwig G. Sekeres, Wilbert Heeringa, Wietse de Vries, Oscar Yde Zwagers, Martijn Wieling, and Goffe Th. Jensma. 2024. Developing infrastructure for low-resource language corpus building. In *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024*, pages 72–78, Torino, Italia. ELRA and ICCL.

Yingli Shen, Wen Lai, Shuo Wang, Xueren Zhang, Kangyang Luo, Alexander Fraser, and Maosong Sun. 2025. DCAD-2000: A multilingual dataset across 2000+ languages with data cleaning as anomaly detection. *arXiv preprint arXiv:2502.11546*. https://huggingface.co/datasets/openbmb/DCAD-2000.

Gary F. Simons, Abbey L. L. Thomas, and Chad K. K. White. 2022. Assessing digital language support on a global scale. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4299–4305, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Atnafu Lambebo Tonja, Israel Abebe Azime, Tadesse Destaw Belay, Mesay Gemeda Yigezu, Moges Ahmed Ah Mehamed, Abinew Ali Ayele, Ebrahim Chekol Jibril, Michael Melese Woldeyohannis, Olga Kolesnikova, Philipp Slusallek, Dietrich Klakow, and Seid Muhie Yimam. 2024. EthioLLM: Multilingual large language models for Ethiopian languages with task evaluation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6341–6352, Torino, Italia. ELRA and ICCL.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Daniel Yaqob. 1997. Transliteration on the Internet: the case of Ethiopic. In *Proceedings of the International Symposium on Multilingual Information Processing*, Tsukuba, Japan.

A. Common Scrawl Web Text Filtering

Here we describe the filtering that we applied to the MADLAD noisy document collection before mining for romanized Tigrinya. We do not start from the MADLAD "clean" collection as this set has already been filtered at the document level by a language identification model that does not cover romanized Tigrinya (Kudugunta et al., 2023). In other words, much of the web text we are interested in has already been removed from MADLAD "clean".

Text selection occurs in several stages. First, some subset of documents are excluded based on broad document-level constraints. Second, newline-delimited paragraphs are filtered to ensure they are mostly Latin script and not from already highly resourced languages. Finally, a series of filters are designed to remove non-linguistic and lower quality (e.g., boilerplate) text. By applying these filters, we aim to produce a corpus of natural text from underrepresented languages, which are typically excluded by pipelines that heavily rely on language ID filtering.

Next we describe the document- and paragraph-level filters, and then present a manual validation of the filters targeted at non-linguistic and low quality paragraphs.

A.1. Document-level Filters

Entire documents containing either (1) a copyright line, (2) some number of low-quality indicator phrases (e.g., indicating gambling or pornography), or (3) *Lorem Ipsum*-like strings were removed before subsequent processing. The presence of such indicators in a document was deemed sufficient to remove the entire document from consideration.

A.2. Paragraph Filter Descriptions

We segment retained documents naïvely by newline delimiter, and consider each of these text blocks to be a paragraph.

Paragraph filters are presented in the order that they were applied. The first four filters ensure that the paragraph is both (a) not labeled by a language identification system as one of a set of high resource languages; and (b) that the sample consists primarily of Latin script words likely encoding language. Other filters attempt to remove low quality text that is unlikely to correspond to spontaneous natural language production, e.g., spam or boilerplate text, non-linguistic text, and general noise.

A.2.1. High-resource language removal

Our focus is in mining underrepresented text, so paragraphs that were confidently labeled as belonging to common Latin script languages were discarded. Text labeled by a proprietary language ID engine with the following ISO-639-2 codes were removed: `af`, `az`, `ca`, `cs`, `da`, `de`, `en`, `es`, `et`, `eu`, `fi`, `fil`, `fr`, `gl`, `hr`, `hu`, `id`, `is`, `it`, `la`, `lt`, `lv`, `ms`, `nl`, `no`, `pl`, `pt`, `ro`, `sk`, `sl`, `sq`, `sv`, `tr`, `uz`, or `vi`. For the current paper, paragraphs labeled as one of the above languages with confidence greater than 0.7 were discarded.

A.2.2. Minimum word count

Paragraphs with less than a minimum count (in this paper, 5) of whitespace-delimited words, each consisting of at least one non-punctuation, non-digit Latin script character were discarded.

A.2.3. Minimum word percentage

Paragraphs with less than a minimum percentage (in this paper, 90%) of whitespace-delimited words containing at least one non-punctuation, non-digit Latin script character were discarded.

A.2.4. Maximum non-Latin script

Paragraphs with more than a maximum percentage (in this paper, 10%) of characters falling outside of the Latin script (ignoring whitespace and punctuation) were discarded.

A.2.5. Standard word form count

We label whitespace-delimited words "standard form" if they meet all of the following conditions: (1) do not begin with a hashtag symbol; (2) have at least 50% of their characters in the Latin script, not counting punctuation or digits; and (3) have standard casing patterns, i.e., either all lowercase, all uppercase or initial letter only capitalized.

Paragraphs with less than a minimum count (in this paper, 3) of "standard form" whitespace-delimited words were discarded.

A.2.6. Standard word form percentage

Following the definition of "standard form" words above, paragraphs with less than a minimum percentage (in this paper, 30%) of "standard form" whitespace-delimited words were discarded.

A.2.7. Maximum word length

Paragraphs containing one or more whitespace-delimited words that consist of more than a maximum character count (in this paper, 100) were discarded.

A.2.8. Maximum mixed word length

Define a "mixed" word as a whitespace-delimited word with both Latin script letters (not punctuation or digits) interspersed with one or more punctuation, digits or non-Latin-script characters. Paragraphs containing one or more mixed words that consist of more than a maximum character count (in this paper, 30) were discarded.

A.2.9. Looks like list

Paragraphs that appeared to consist of a list of relatively short, delimited items were discarded.

A.2.10. Looks like code

Paragraphs that appeared to be code or are heavily marked-up were discarded.

A.2.11. Looks like Lorem Ipsum

Paragraphs that matched on *Lorem Ipsum* strings were discarded. Note that there was also a document-level constraint applying similar filtering to discard whole documents.

A.2.12. Maximum paragraph length

Paragraphs consisting of more than a maximum number of Unicode code points (for this paper, 36,000) were discarded.

A.2.13. Maximum length between delimiters

Paragraphs containing spans without any text delimiters (e.g., punctuation such as comma or period) consisting of more than a maximum number of Unicode code points (for this paper, 4,100) were discarded.

A.2.14. Boilerplate / low quality

Paragraphs that match repeatedly on words from a small manually curated vocabulary associated with broad classes of low quality (e.g., spammy) web content were discarded. Note that there was also a document-level constraint applying similar filtering to discard whole documents.

A.2.15. ACTG percentage

Paragraphs with more than a maximum percentage (in this paper, 90%) of whitespace-delimited words consisting only of the characters ACTG were discarded. These generally represent biosequences. ACUG is also tested, to cover RNA sequences.

A.2.16. Maximum alphanumeric words

Define an "alphanumeric" word as a whitespace-delimited word with both digits and letters but no punctuation, e.g., C3PO. Paragraphs with more than a maximum number of alpha-numeric words (in this paper, 100) without any punctuation were discarded.

A.2.17. Percentage in top length bin

Let $\hat{L}$ be the maximum length in unicode code points of any whitespace delimited-word in a paragraph. For paragraphs with more than some minimum number of words (in this paper, 100), if the percentage of words in the paragraph with length $\hat{L}$ is greater than some maximum percentage (in this paper, 95%), the paragraph was discarded.

A.2.18. Minimum type/token ratio

For paragraphs with more than some minimum number of whitespace-delimited words (in this paper, 100), if the ratio of the number of word types in the paragraph to the number of word tokens in the paragraph falls below some minimum (in this paper, 0.02), the paragraph was discarded.

A.2.19. Repetitive paragraph pattern

For paragraphs with more than some minimum number of whitespace-delimited words (in this paper, 10), if either (1) every other word – even or odd – is the same word; or (2) every word consists of the same single character, the paragraph was discarded.

A.2.20. Pct words w/same first character

For paragraphs with more than some minimum number of whitespace-delimited words (in this paper, 10), if the percentage of words in the paragraph that start with the same initial character (regardless of character identity) rises above some maximum (in this paper, 95%), the paragraph was discarded.

A.2.21. Pct words w/same first symbol

For paragraphs with more than some minimum number of whitespace-delimited words (in this paper, 10), if the percentage of words in the paragraph that start with the same initial punctuation or non-alphanumeric symbol (e.g., @ or #) rises above some maximum (in this paper, 80%), the paragraph was discarded.

A.2.22. Hashtag word and symbol

Paragraphs that include the substring "hashtag" and also have whitespace-delimited words beginning with a hashtag (#) were discarded.

A.2.23. Standard form language removal

Using the definition of a "standard form" whitespace-delimited word given in Section A.2.5, we rerun the language identification system on the paragraph with only standard form words retained. Using the same confidence threshold and list of languages shown in Section A.2.1, paragraphs that are labeled as a language in the list were discarded.

A.2.24. URL affix percentage

Paragraphs with more than a maximum percentage (in this paper, 30%) of whitespace-delimited words either prefixed by "www." or suffixed by "'.com"' were discarded.

A.2.25. Max pct single character words

Paragraphs with more than a maximum percentage (in this paper, 90%) of whitespace-delimited words consisting of a single character were discarded.

A.2.26. Boilerplate prefix or suffix

Paragraphs with a prefix substring from a curated list of prefixes associated with boilerplate text, or a suffix substring from a curated list of boilerplate-associated suffixes, were discarded.

A.2.27. List of times

Paragraphs that appeared to be a list of simple time values were discarded.

Section describing filter constraint	Round 1 Agr / N	Round 2 Agr / N	Round 3 Agr / N	Overall Agr / N	Prec
Section A.2.5: Standard word form count	1 /1	5 /5	9 /9	15 /15	1.00
Section A.2.6: Standard word form percentage	3 /3	9 /9	16 /16	28 /28	1.00
Section A.2.7: Maximum word length	5 /6	5 /5	4 /4	14 /15	0.93
Section A.2.8: Maximum mixed word length	4 /6	9 /9	10 /11	23 /26	0.88
Section A.2.9: Looks like list	6 /6	16 /16	18 /18	40 /40	1.00
Section A.2.10: Looks like code	6 /6	8 /8	8 /8	22 /22	1.00
Section A.2.11: Looks like Lorem Ipsum	0 /1	5 /5	9 /9	14 /15	0.93
Section A.2.12: Maximum paragraph length	0 /0	0 /0	0 /0	0 /0	
Section A.2.13: Max length between delimiters	3 /4	5 /5	6 /6	14 /15	0.93
Section A.2.14: Boilerplate / low quality	1 /1	7 /7	9 /9	17 /17	1.00
Section A.2.15: ACTG percentage	1 /1	5 /5	9 /9	15 /15	1.00
Section A.2.16: Max alphanumeric words	0 /0	5 /5	10 /10	15 /15	1.00
Section A.2.17: Percentage in top length bin	2 /2	10 /10	15 /15	27 /27	1.00
Section A.2.18: Minimum type/token ratio	5 /5	9 /9	8 /8	22 /22	1.00
Section A.2.19: Repetitive paragraph pattern	4 /4	9 /9	17 /17	30 /30	1.00
Section A.2.20: Pct words w/same 1st char	6 /6	12 /12	20 /20	38 /38	1.00
Section A.2.21: Pct words w/same 1st symbol	5 /5	6 /6	6 /6	17 /17	1.00
Section A.2.22: Hashtag word and symbol	3 /3	5 /5	7 /7	15 /15	1.00
Section A.2.23: Standard form lang removal	13 /13	29 /29	44 /44	86 /86	1.00
Section A.2.24: URL affix percentage	2 /2	6 /6	8 /8	16 /16	1.00
Section A.2.25: Max percent single char words	4 /4	8 /8	11 /11	23 /23	1.00
Section A.2.26: Boilerplate prefix or suffix	8 /8	6 /6	2 /2	16 /16	1.00
Section A.2.27: List of times	4 /4	5 /5	6 /6	15 /15	1.00
Section A.2.28: Gzip compression score	10 /10	17 /17	17 /17	44 /44	1.00
Section A.2.29: Maximum repeated char length	5 /5	6 /6	8 /8	19 /19	1.00
Section A.2.30: Minimum language substring	43 /44	82 /82	102 /102	227 /228	1.00
Section A.2.31: Maximum diacritic percentage	2 /2	5 /5	8 /8	15 /15	1.00

Table 3: Precision of filtering constraints according to the authors' consensus annotation. *Agr*: number agreed with consensus, *N*: number of examples, and *Prec*: precision. Precision is only given for the union over all rounds. Note that these examples were sampled after document-level filtering (e.g., by copyright notice and document-level boilerplate detection). Note that a single paragraph may violate multiple constraints, thus the counts do not sum to 413 (the number of sampled paragraphs that violated at least one constraint).

A.2.28. Gzip compression score

Let s be the size of the paragraph after using gzip, and let b be the size after gzip of a baseline string of the same length as the paragraph. If $ks < b$ for some k (in this paper, $k = 3$), then the paragraph was discarded. In other words, paragraphs with very high compression rates for their length were discarded.

A.2.29. Maximum repeated char length

After removing whitespace and converting to lower case, let m be the length of the longest substring consisting of a single repeated character. Paragraphs with m longer than some maximum length (in this paper, 30) were discarded.

A.2.30. Minimum language substring

Using a sliding window, language identification is applied at each whitespace-delimited word in the paragraph, providing a label at each word. Let m be the longest string of words in the paragraph labeled with the same language label, where that label is not in the list of languages to exclude from Section A.2.1. If m is less than a minimum length (in this paper, 3), the paragraph was discarded.

A.2.31. Maximum diacritic percentage

Paragraphs with more than a maximum percentage (in this paper, 95%) of characters with diacritics were discarded.

Instructions:
For each sample text in column A in the spreadsheet below, please indicate Yes in column B if the example is clearly Tigrinya, even if the style of writing in the Latin script is not how you would write it. If the sample text has some loan words from another language but the main language is Tigrinya, then still mark that as Yes. If the sample text is not Tigrinya, or you cannot be certain, indicate No in the drop down menu in column B. If you indicate No in column B and can easily identify the actual language of the sentence, please put that information in column C. You do not need to spend much time on this part, only if it is obvious to you. Feel free to comment on any feature of the text in column D. For example, if the sample text is not actually language, or contains questionable content of some sort.

Below are four examples. The first is indeed romanized Tigrinya, the other three are not Tigrinya. The language is identified in the second example but not in the third or fourth. Comments to the effect that the sample is not language is added to the fourth example.

sample text	**Tigrinya? Y/N**	**If N, identify the language if you can.**	**Any comments on text (e.g., spam/boilerplate/porn or otherwise low quality)**
entay aynet tseweta eiki kt'tsaweti tdeli?	Yes		
yetesegnut tshufochna yeleloch berkata srawoch balebet nacehw	No	Amharic	
jagat to uske ek ansh matra men hai.	No		
hck hdk hdk hdt hdt hdv hdv hen hen hew hft hft hgs hgs hgu hgu	No		Not Language

Table 4: Instructions for Tigrinya manual annotation, along with four examples provided to annotators.

A.3. Filter Validation

Filter validation consisted of three rounds of annotation by the paper authors: the first round of 100 paragraphs was annotated by three of the authors, and the second and third rounds (168 and 272 paragraphs, respectively) were annotated by two of the authors. Authors annotated whether they believed each paragraph should be *kept* or *discarded* by an ideal pipeline. Paragraphs were labeled discardable if the text consisted of non-natural or boilerplate language, mostly consisted of head language text, or formulaic (e.g., list of items). Overall, 150 out of a total of 540 sampled paragraphs (27.8%) satisfied all of the pipeline constraints.

From the set of paragraphs filtered by each stage in the pipeline (and the final set of documents that passed through all stages without being filtered), a target of 100 examples were extracted using an appropriately set random number generator. These were shuffled and a final random sample taken that ensured some minimum coverage for each filter stage.

After each round of annotation, authors arrived at a consensus annotation after discussion, and the multiple rounds helped codify the annotation guidelines. The three annotators agreed unanimously on 80% of first round examples. In rounds two and three, the two annotators agreed on 94% and 94.5% of examples before arriving at a consensus. For the two annotators that annotated through all three rounds, these corresponded to agreement according to Cohen's Kappa (Cohen, 1960) of 0.66, 0.82, and 0.86 for each of the respective rounds – 0.81 overall, indicating strong inter-annotator agreement. Below we give the confusion matrix between consensus annotation (column) and whether the paragraph was filtered by the pipeline (row) across all rounds, *discarded* vs. *kept*.

384	6
29	121

Overall, the estimated precision of our constraints is 95.3%, with recall of 80.7%. Bear in mind that we oversampled examples that satisfied all constraints. Precision of each of the paragraph-level filters is given across each of the annotation rounds in Table 3. While the counts are quite low for many filters, the precision is almost always 100%, with a minimum precision of 88%, e.g., the `longest_mixed_word` constraint.

B. Tigrinya Annotation Instructions

Table 4 contains the instructions provided to each of the Tigrinya annotators, along with four example texts.

C. South Asian Experiment Variance

Table 5 presents the means and standard deviations of minimum character error-rate percentage (mCER%) over 10 trials in each condition, as reported in Table 1.

	language		native source	uconv source
script	target	train	mean / stdev	mean / stdev
deva	brx	brx	4.80 / 0.06	5.17 / 0.13
deva	brx	hin	31.06 / 0.86	31.15 / 0.82
deva	brx	kok	38.71 / 3.06	38.53 / 2.97
deva	brx	mai	32.12 / 0.64	32.56 / 0.95
deva	brx	mar	31.63 / 0.35	31.46 / 0.60
deva	brx	nep	33.25 / 0.67	33.68 / 0.63
deva	brx	san	53.64 / 1.17	53.68 / 0.57
deva	hin	brx	29.60 / 0.35	28.27 / 0.36
deva	hin	hin	8.13 / 0.34	7.89 / 0.13
deva	hin	kok	15.09 / 4.16	13.42 / 1.05
deva	hin	mai	11.64 / 0.51	11.47 / 0.50
deva	hin	mar	11.54 / 4.35	9.92 / 0.54
deva	hin	nep	12.01 / 0.79	11.60 / 0.20
deva	hin	san	27.43 / 1.63	27.41 / 1.49
deva	kok	brx	32.02 / 0.47	30.45 / 0.65
deva	kok	hin	13.32 / 1.01	12.81 / 0.52
deva	kok	kok	11.00 / 0.23	10.66 / 0.50
deva	kok	mai	15.41 / 0.66	14.55 / 0.54
deva	kok	mar	11.72 / 0.15	11.56 / 0.29
deva	kok	nep	17.03 / 1.70	15.86 / 0.56
deva	kok	san	18.14 / 1.54	17.80 / 1.04
deva	mai	brx	29.39 / 0.31	28.94 / 0.37
deva	mai	hin	9.94 / 0.69	9.35 / 0.31
deva	mai	kok	12.62 / 3.17	11.46 / 0.61
deva	mai	mai	7.09 / 0.30	6.99 / 0.26
deva	mai	mar	11.50 / 3.23	9.90 / 0.61
deva	mai	nep	11.23 / 0.23	10.84 / 0.28
deva	mai	san	16.36 / 0.79	16.65 / 0.82
deva	mar	brx	24.96 / 0.42	23.35 / 0.74
deva	mar	hin	5.58 / 0.88	5.06 / 0.38
deva	mar	kok	7.29 / 1.71	7.16 / 1.60
deva	mar	mai	8.93 / 0.65	7.78 / 0.74
deva	mar	mar	3.79 / 0.10	3.56 / 0.22
deva	mar	nep	9.36 / 1.86	8.05 / 0.74
deva	mar	san	23.76 / 1.67	23.48 / 1.46
deva	nep	brx	26.21 / 1.39	25.38 / 0.64
deva	nep	hin	8.83 / 0.97	8.14 / 0.31
deva	nep	kok	12.81 / 1.29	11.72 / 1.64
deva	nep	mai	10.82 / 1.28	10.78 / 1.18
deva	nep	mar	9.38 / 0.39	8.53 / 0.47
deva	nep	nep	3.95 / 0.14	3.91 / 0.22
deva	nep	san	23.53 / 1.18	23.55 / 1.29
deva	san	brx	37.16 / 0.62	37.32 / 0.45
deva	san	hin	15.33 / 0.71	13.64 / 0.68
deva	san	kok	11.11 / 3.01	9.54 / 0.77
deva	san	mai	13.85 / 0.66	13.01 / 0.43
deva	san	mar	14.05 / 1.08	13.12 / 0.44
deva	san	nep	15.49 / 0.64	15.19 / 0.80
deva	san	san	3.48 / 0.25	3.55 / 0.14
beng	asm	asm	4.66 / 0.18	5.40 / 0.19
beng	asm	ben	30.72 / 0.80	24.57 / 0.68
beng	ben	asm	21.14 / 2.01	19.45 / 0.23
beng	ben	ben	7.83 / 0.29	7.42 / 0.25

Table 5: Means and standard deviations of mCER% in experiments reported in Table 1.