Improving Informally Romanized Language Identification

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

Abstract

The Latin script is often used to informally write languages with non-Latin native scripts. In many cases (e.g., most languages in India), the lack of conventional spelling in the Latin script results in high spelling variability. Such romanization renders languages that are normally easily distinguished due to being written in different scripts – Hindi and Urdu, for example – highly confusable. In this work, we increase language identification (LID) accuracy for romanized text by improving the methods used to synthesize training sets. We find that training on synthetic samples which incorporate natural spelling variation yields higher LID system accuracy than including available naturally occurring examples in the training set, or even training higher capacity models. We demonstrate new state-of-the-art LID performance on romanized text from 20 Indic languages in the Bhasha-Abhijnaanam evaluation set (Madhani et al., 2023a), improving test F1 from the reported 74.7% (using a pretrained neural model) to 85.4% using a linear classifier trained solely on synthetic data and 88.2% when also training on available harvested text.

1 Introduction

Web crawls are an important source of multilingual training data for natural language modeling, containing rich and diverse spontaneous language, albeit alongside less useful content such as boilerplate and non-language text. Due to this latter noise, to reach adequate levels of data quality, aggressive filtering is typically applied, including setting thresholds on language identification (LID) confidence to retain text (see, e.g., Raffel et al., 2020; Xue et al., 2021; Abadji et al., 2022; Kudugunta et al., 2023; Kargaran et al., 2024).

Filtering text by LID confidence places languages that are less confidently identified at a higher risk of being filtered out, and, indeed, certain classes of text are heavily underrepresented in such collections as a result, such as the romanized text detailed below. This can become a self-reinforcing status, to the extent that such collections may be used as sources of training data for subsequent LID systems. For example, informal romanization – the use of the Latin script without orthography to write languages that are natively written with non-Latin scripts – is both common, e.g., in the languages of South Asia, and generally makes the languages much more difficult to identify. The 22 scheduled languages of India all use non-Latin scripts in their official writing systems, but are also commonly written informally in the Latin script (Brandt, 2020). Even so, romanized texts in these languages are mostly omitted from large, multilingual web corpora such as multilingual C4 (mC4, Xue et al. 2021),[1] and are relatively sparse in MADLAD-400 (Kudugunta et al., 2023)[2] and GlotCC (Kargaran et al., 2024).[3]

The distinction between LID system performance on text in the native scripts of languages versus romanized text in those languages is strikingly demonstrated in the Bhasha-Abhijnaanam (denoted B-A) LID benchmark (Madhani et al., 2023a)[4] for the 22 scheduled languages of India. The best systems reported in that paper achieved nearly 99% accuracy on the native script LID task, while the best reported performance on the romanized task (covering 20 of the 22 languages) reached just over 80% accuracy. Further, to achieve this latter result, they relied on a relatively expensive BERT model (Devlin et al., 2019); a simple fast-

[1] mC4 contains romanized text in 6 languages that natively use other scripts, but only one South Asian language was included (Hindi), and quality assessments of the romanized sets in mC4 have been unfavorable (Kreutzer et al., 2022).

[2] https://huggingface.co/datasets/allenai/MADLAD-400, data released under CC-BY-4.0 license.

[3] https://github.com/cisnlp/GlotCC, data released under CC0-1.0 license. See Table 3 for data sizes.

[4] https://huggingface.co/datasets/ai4bharat/Bhasha-Abhijnaanam, data released under CC0 license.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 2319–2337
November 4-9, 2025

Text (Joulin et al., 2016, 2017) linear model yielded the best performance on native script LID, but on the romanized task its performance fell far behind the best system at just 71.5% accuracy.

Given the dearth of natural, collected training data in the Latin script for these languages, the above romanized LID results were achieved using synthesized training data created by automatically romanizing native script training sets using a neural sequence-to-sequence transliteration model (Madhani et al., 2023b). In this paper, we evaluate methods for improving romanized LID system performance, including synthesizing training data in a way that mimics natural spelling variation in romanized text; and augmenting synthesized data with distantly supervised datasets (MADLAD-400 and GlotCC). Through careful controlled experimentation, we demonstrate that:

- Synthesizing training data with spelling variation by sampling from romanization models provides very large accuracy improvements for this task, even when using smaller, less accurate romanization models;
- These improvements are obtained to a larger extent by lightweight linear models, so that higher capacity pretrained models are not needed to reach the best reported results; and
- Augmenting the training data by including diversely synthesized copies of the training set and/or some independently harvested data yields further modest improvements.

Our best system reduces the best previously reported error rate on this task by over 60% relative.[5]

We also provide extensive analysis of language confusions, system errors and the kinds of spelling variation produced by our synthesis methods.

2 Background and preliminaries

2.1 Language identification (LID) systems

LID is a task with a long history, dating from at least Mustonen (1965). Jauhiainen et al. (2019)'s survey notes that LID is typically cast as a text classification task over a closed set of mutually exclusive classes.[6] They also note the early adoption of character n-grams as features in diverse LID classification approaches (Church, 1986; Beesley, 1988; Cavnar and Trenkle, 1994), which remains a key feature set for modern LID (Lui and Baldwin, 2012; Brown, 2014; Kargaran et al., 2023; Burchell et al., 2023) and dialect identification systems (Çöltekin et al., 2018; Baimukan et al., 2022).

Document level LID assigns labels to entire documents, within which there is typically redundant evidence, hence the classification task is easier. Shorter samples, including sub-sentential strings, present more of a challenge, but such systems can be applied in more scenarios than document-level systems, including, e.g., identifying different language spans in multilingual documents.

As stated in the introduction, LID systems are often used for analyzing or filtering large text collections, hence efficient inference is a key consideration. Generally speaking, feature-based linear classifiers are practical alternatives to more computationally expensive neural methods (Toftrup et al., 2021; Adebara et al., 2022), including those that might be based on pretrained language models (Mohan et al., 2023; Manukonda and Kodali, 2025). For challenging scenarios, such as when two languages are highly confusable, more expressive classifiers can yield accuracy improvements that justify the extra computational expense.

While modern neural language models have been shown to perform well on general LID tasks across a range of world languages, their performance depends on the subset of languages the model is prompted to identify and the scripts these languages are written in (Chen et al., 2024). More importantly, such models are extremely compute and memory-intensive to be deployed at scale, e.g., at the corpus filtering stage. In fact, Kargaran et al. (2023) explicitly cites inference throughput as a consideration for developing the GlotLID language identification system, which was used to filter the GlotCC corpus mentioned earlier.

2.2 LID of romanized language

LID systems based on lightweight linear models typically rely on features consisting of short collocations — character n-grams — to distinguish between languages. When two languages use different scripts, which are encoded in distinct Unicode blocks, even character unigrams can be sufficient to distinguish the two. Written in their native scripts, Chinese and English are straightforward to distinguish, hence are unlikely to be confused by

[5] Explicit system details beyond those provided in this paper along with any new resources and scripts required to replicate these results are available at https://github.com/google-research/google-research/tree/master/informally_romanized_lang_id.

[6] Multilingual text, which would seem to contradict this idea of mutual exclusivity, is composed of shorter monolingual text spans for which a single tag per span would suffice.

Hindi	Urdu	Attested romanizations
संपूर्ण	سمپورن	sampoorn, samporan, sampuran, sumpoorn

Table 1: Human-attested romanizations of a word in Hindi and Urdu that translates as 'complete" in English. These attested romanizations are shared across both Hindi and Urdu.

a classifier; similarly Hindi and Urdu – the former written in the Devanagari script, the latter in Perso-Arabic – are quite easy to distinguish given the lack of character overlap. However, when Hindi and Urdu are written in the Latin script (romanized), not only do they share the same characters, but the character collocations observed in the two highly mutually-intelligible languages will greatly overlap. For example, Table 1 presents an example of a word shared by Hindi and Urdu with multiple overlapping attested romanizations in both languages.

Accurate LID of romanized text is of practical importance as the Latin script is often used as an informal medium between speakers of languages that are formally written in a different script, e.g., in text messages or social media posts (Ahmed et al., 2011; Irvine et al., 2012; Adouane et al., 2016; Baruah et al., 2024; Perera et al., 2025). The lack of (or limited access to) standard spelling conventions for these languages in the Latin script is due to multiple confounding sociolinguistic, technological and political factors at play in South Asia (Choksi, 2020; Brandt, 2020) and beyond (Bahri, 2022), and poses yet another challenge for romanized LID. Additional practical uses of romanized LID include improving romanized text training data coverage for modern neural language models (Caswell et al., 2020), LID of romanized text entry in mobile keyboards (Wolf-Sonkin et al., 2019), extending neural methods to poorly documented languages (Post and Burling, 2017; Aji et al., 2022) or languages without orthography (Torwali, 2020), among others.

Several datasets have been released that contain parallel native/Latin script data, which permits, among other things, training and/or validation of transliteration models (see Section 2.4). These include the Dakshina dataset (Roark et al., 2020), which includes single word native/Latin script pairs in romanization lexicons as well as native/Latin script parallel full sentences, in addition to native-script-only Wikipedia text, for twelve South Asian languages. The Aksharantar dataset (Madhani et al., 2023a,b) contains mined single word native/Latin script pairs in romanization lexicons for 21 South Asian languages. And, of course, the benchmark being used for this work, the B-A dataset, also contains parallel resources of this sort.

Literature on romanized LID *per se* is relatively scarce. In one of the earliest works on the subject, Pavan et al. (2010) propose a string similarity-based system for identification of romanized documents in Hindi, Telugu, Tamil, Kannada and Malayalam. More recently, Nielsen et al. (2023) proposed approaches for distinguishing romanized Hindi from romanized Urdu in the Dakshina dataset. Dey et al. (2024) compared support vector machines (SVM, Cortes and Vapnik, 1995) and finetuned XLM-RoBERTa architecture (Conneau et al., 2020) for romanized LID using data from 12 South Asian languages sourced from the B-A and Aksharantar datasets. Beyond South Asia, Adouane et al. (2016) investigated LID of romanized Arabic and Berber using SVM classifiers trained on word- and character-level n-gram features.

2.3 Classifiers

We consider two classes of LID models in this paper: fastText linear models (Joulin et al., 2016, 2017)[7] and neural pretrained transformer-based multilingual T5 (mT5) models (Xue et al., 2021).[8]

While linear classifiers have low inference latency, pretrained mT5 classifiers are more expressive, and may be able to better disambiguate similarly written languages. Since prior results demonstrated that pretrained neural models outperformed linear models (Madhani et al., 2023a), we use mT5 to gauge whether lightweight methods can provide competitive results to such models, and to investigate the interaction between training set and classifier capacity.[9] See Appendix A for training details.

2.4 Transliteration models

Transliteration of text from one script to another is typically framed as a sequence-to-sequence task. While this task can be performed with neural sequence-to-sequence models, non-neural methods are also competitive. Kirov et al. (2024) recently examined romanization (transliteration from native to Latin script) methods for synthesizing full sentence parallel text, and evaluated the character error rate (CER) achieved by multiple model types.

[7] https://fasttext.cc/, MIT license.

[8] https://github.com/google-research/multilingual-t5, Apache License 2.0.

[9] Also, mT5 pretraining is unlikely to be contaminated with the training or validation data for the current task, since the model predates the task. Additionally, unlike modern LLMs, it has not been instruction-tuned. For these reasons, it provides an informative comparison within our controlled experiments.

For single, non-ensembled models producing single best romanizations, CER[10] ranged from 2.6% from a fully pretrained T5 model to 3.7% for a non-neural pair n-gram method; these results were slightly improved via further system ensembling. A more complex evaluation of k-best outputs showed a similar pattern of relatively narrow ranges of error rates. In this paper, we use non-neural pair n-gram methods for automatic romanization, trained following the approach outlined in Kirov et al. (2024).

Briefly, pair n-gram models (Bisani and Ney, 2008) are a class of models used to map between two sets of discrete token sequences, and are commonly used for transliteration (Hellsten et al., 2017; Kirov et al., 2024). They can be directly encoded as weighted finite-state transducers (WFSTs), which permit efficient exact inference. Given a parallel corpus of input/output strings, expectation maximization (EM) is used to create strings of aligned single character input/output pairs, from which n-gram models are trained. For example, if `ABC` is aligned with `abc`, then the EM alignment algorithm may produce `A:a B:b C:c`, and this string (along with a full corpus of other such alignments) are used to train an n-gram model. The pair symbols are then split into input and output symbols, so that the n-gram model is encoded as a transducer. See Kirov et al. (2024) for details on training such models from romanization lexicons in the Dakshina dataset (Roark et al., 2020), which we also use.[11]

The LID benchmark that we investigate in this paper provided romanized training and development data synthesized using IndicXlit (Madhani et al., 2023b), a neural sequence-to-sequence transliteration model trained on over twenty south Asian languages from the Aksharantar romanization lexicon, a lexicon we also use in this paper.[12]

3 Methods

3.1 Evaluation

We evaluate classifiers on the B-A romanized LID task (Madhani et al., 2023a). The benchmark provides training/development data for each language in the native script, and the IndicXlit system, described in Section 2.4, was used to romanize the native script text to also provide synthetic training/development data in the Latin script for 20 of the 22 languages.

The B-A romanized test set consists of full sentence examples from Dakshina (Roark et al., 2020) for the following 11 languages: Bangla, Gujarati, Hindi, Kannada, Malayalam, Marathi, Punjabi, Sindhi, Tamil, Telugu, and Urdu. Each of these languages contains between 4,371 to 4,881 test examples. The additional languages in the romanized test set (Assamese, Bodo, Kashmiri, Konkani, Maithili, Manipuri, Nepali, Oriya, and Sanskrit)[13] each contain roughly 10% as many test examples as the Dakshina set (423 to 512 examples per class).

For all runs, languages in the training set are oversampled to the plurality class – the language with the most examples in a given training set – to ensure a uniform prior distribution over languages. For instance, if our training set contains 1,000 Hindi, 750 Bangla, and 500 Bodo examples, we oversample the Bangla and Bodo examples to yield 3,000 total training examples, by duplicating 250 Bangla and 500 Bodo examples. We train fastText models from scratch on the various training sets and also use these training sets to finetune the pretrained public base and large mT5 checkpoints.

3.2 Development set

Note that, while the synthetic dev set provided by the B-A benchmark can be useful for tuning some system parameters, it has a critical mismatch with the test set: the test set consists of human romanized text, with varied romanizations; but the synthetic dev set consists of machine generated romanizations without such variation. In particular, the IndicXlit system romanized each native script word the same way whenever it was encountered.

In addition to evaluating on real (human) romanized text, Madhani et al. (2023a) also evaluated their system's LID performance when applied to collections of synthesized romanizations, to assess the importance of the mismatch between romanizations used for training and evaluation. They show that LID accuracy improves from 80.4% to 96.0% when evaluating on human versus synthetic romanizations. This is unsurprising, since the distribution of synthetic romanizations are (for obvious reasons) far more similar to the training/dev sets than human romanizations are. Thus, rather than use the provided synthetic dev set, we instead chose to employ a human romanized dev set for a subset of

[10] CER was assessed against the closest match from multiple references, hence is a minimum CER over that set.

[11] https://github.com/google-research-datasets/dakshina, data released under CC BY-SA 4.0 license.

[12] https://huggingface.co/datasets/ai4bharat/Aksharantar, data released under CC-BY and CC0 licenses.

[13] The romanized benchmark omits Dogri and Santali.

		Sentence in the native Devanagari script:									
System	method	इनमें	33	स्कूल	पिछले	साल	ही	बंद	हुए	हैं	।
B-A IndicXlit	Neural	inmein	33	school	pichhale	saal	hii	bund	hue	hain	.
Dakshina 2-gram	Pair n-gram	inamen	33	scool	pichale	sal	hi	bnd	hue	han	.
Dakshina 3-gram	Pair n-gram	inmein	33	scool	pichle	sal	hi	band	hue	hain	.
Dakshina 4-gram	Pair n-gram	inmen	33	school	pichhale	sal	hi	band	hue	han	.
Aksharantar 3-gram	Pair n-gram	inmen	33	school	pichhle	sal	hi	band	huye	han	.
	English translation:	Of these, 33 schools were closed last year alone.									

Table 2: Romanizations sampled from different transliteration models.

the B-A languages, which we describe here.

The Dakshina dataset (Roark et al., 2020) consists of three parts for each of the languages in the collection: single word romanization lexicons, where native script words are paired with one or more attested romanizations;[14] native script Wikipedia text samples; and 10k human romanized full sentences sampled from a subset of the native script Wikipedia text. These latter full sentence romanizations are split into development and test partitions of 5k sentences each, and the B-A test set includes the 5k sentences from the test partition. This leaves the 5k development partition for use as a development set, covering an eleven (out of twenty) language subset of the B-A task.

All hyperparameters were first tuned on this Dakshina development set, restricting the training set to the eleven languages shared with Dakshina, sampled without replacement to 1,000 examples per language. See Appendix E for additional discussion and results comparing the synthetic dev set with this human romanized dev set.

3.3 Romanized training set synthesis

The B-A synthesized training set for the romanized LID task was created by automatically romanizing the native script training set. In this paper, we explore other methods for training set synthesis from the same native script training set, using pair n-gram models of various Markov orders (2–4), as described in Section 2.4. These models were trained on reference input/output word pairs either from the Dakshina (Roark et al., 2020) or Aksharantar (Madhani et al., 2023b) romanization lexicons. In fact, significant portions of the Dakshina lexicons are directly included in the Aksharantar lexicons. However, these Dakshina subsets within Aksharantar have greater typical fertility than the rest of Aksharantar, i.e., more attested romanizations per word. Over the 11 languages for which there are Dakshina romanization lexicons, the Aksharantar romanization lexicons contain just over 18 million unique native script words, but with just 1.011 romanizations per word. For these same languages, the Dakshina training lexicons include 265,000 unique native script words (25k in all but Sindhi, for which there are 15k), which is less than 1.5% of Aksharantar; however, they contain on average 2.8 romanizations per word,[15] hence provide many more attested romanizations per word.

Trained transliteration models are used to synthesize training sets for romanized LID from the B-A native script LID training data. Thus the source native script data is identical across the different synthesis methods, including the synthesized romanized training set released in the B-A benchmark. Table 2 presents romanizations for a Hindi sentence sampled from different transliteration models.

We use two different methods to romanize a word given a model. First, we simply take the best scoring romanization of the given word, the *1-best* decoded romanization. Alternatively, we generate top-k romanizations and *sample* over a renormalized distribution of these, conditioned on the native script string. For this study, we extract the global 8-best romanizations and their probabilities using the Viterbi algorithm. Such exact global inference is possible since the romanization models are encoded as weighted finite state transducers. In this sense, we are sampling with temperature 1 over the set of global 8-best romanizations. Note that, when sampling from the k-best romanizations, each pass over the training data yields another distinctly romanized corpus. Hence, we also explore producing multiple distinctly-romanized copies of the training data. Functionality for performing such sampling is built into the transliteration utilities of the Nisaba library, see the URL in Footnote 5 for details.

Both Dakshina and Aksharantar romanization lexicons omit some common symbols from their

[14]These romanization lexicons and others are used to train romanization models used for training data synthesis.

[15]The Dakshina lexicons contain between 1.7 and 4.2 romanizations per word on average, depending on the language.

Language	MADLAD noisy	MADLAD clean	GlotCC
Assamese			2.0K
Bengali	1.3M	12.0K	2.4K
Gujarati	688.4K	5.4K	285
Hindi	22.6M	1.2M	32.4K
Kannada	766.0K	10.1K	416
Konkani			8.8K
Malayalam	3.5M	77.3K	3.5K
Manipuri			1.0K
Marathi			1.7K
Nepali			968
Oriya			494
Punjabi			4.7K
Sindhi			30
Tamil	3.4M	142.7K	4.5K
Telugu	4.4M	269.1K	16.1K
Urdu			84.6K

Table 3: Number of sentences for each B-A romanized language in GlotCC and MADLAD. Where there is no number, there are no sentences. Bodo, Kashmiri, Maithili and Sanskrit have no romanized examples in any of these datasets.

training data, including things like native script digits and punctuation, which are romanized in the baseline synthesized training data. From this data, we identified romanized tokens that fell outside of the coverage of Dakshina or Aksharantar, and included these tokens in model training to provide equivalent coverage of the training set.

When generating synthetic training data for all 20 languages in the B-A LID task, we make use of Dakshina-trained transliteration models for those languages covered by Dakshina, and Aksharantar-trained transliteration models for the rest. We find that the spelling variation induced by drawing samples from transliteration models does indeed reflect the kind of natural spelling variation reported in the literature on informal South Asian romanization. Phenomena such as implicit vowel realization, variation in indicating vowel quality, and gemination are some of the most frequent variations in these samples. See Section 4.7 and Appendix D for analysis of spelling variability in synthetic samples.

3.4 Harvested natural examples

Another source of romanized examples for training LID systems is already harvested text – which we have noted is relatively sparse, but is still important to assess as a source of distant supervision. We consider two potential sources of harvested natural romanized text. The first is MADLAD-400, a filtered subset of Common Crawl that covers a wider range of languages than mC4. The release contains text for 7 of the 20 romanized languages in B-A, with both 'noisy' and 'clean' sets for each. The second is GlotCC, a corpus of web documents whose language has been identified with high confidence by GlotLID, a wide coverage language identification model. Table 3 lists the number of natural example sentences in each of these corpora for each of the B-A evaluation set languages.

Training set		Classifier F1 fastText	mT5-base	mT5-large
B-A training		81.4	**84.9**	**85.3**
Dakshina	2-g 1-best	72.5	75.1	79.2
	3-g 1-best	80.6	80.4	81.7
	4-g 1-best	**83.0**	80.9	82.6
Aksharantar	3-g 1-best	78.3	79.6	78.7

Table 4: Dakshina development set performance as a function of training set and model class, comparing baseline B-A synthesized training data with other synthesis methods that romanize every instance of a word the same (1-best).

4 Results

In this section, we step through results on the Dakshina (human romanized) development set as we change the LID training set by using different training set synthesis models and different methods for romanizing with those models. In each trial, we present the LID F1 score for fastText, mT5-base and mT5-large classifiers trained on the version of the training set associated with each trial, always comparing against training on the baseline training data that came with the B-A dataset.[16]

The results are structured so that the relative contribution of different key methods are demonstrated on the development set. We begin with results using 1-best romanizations (Section 4.1), then move on to demonstrate improvements due to sampling (Section 4.2). We follow this by examining the impact of combining distinctly synthesized datasets (Section 4.3), sampling multiple copies of a dataset with the same synthesis method (Section 4.4), and making use of distantly supervised corpora (Section 4.5). We then present selected system performance on the test set, and finish the section with some discussion and analysis.

4.1 Best romanizations

Table 4 compares systems trained on the baseline training set with those synthesized using pair n-gram transliteration models of various orders, themselves trained on either the Dakshina or Aksharan-

[16]We also computed accuracy for all trials, but do not report it for development set results as this metric was very similar to F1 for those trials.

Training set		Classifier F1 fastText	mT5-base	mT5-large
B-A training		81.4	84.9	85.3
Dakshina	2-g sampled	87.0	85.4	87.6
	3-g sampled	88.0	86.3	87.3
	4-g sampled	**88.5**	**87.5**	**88.4**
Aksharantar	3-g sampled	85.6	84.0	84.9

Table 5: Dakshina development set performance as a function of training set and model class, comparing baseline B-A synthesized training data with other synthesis methods that sample romanizations from k-best output, so that words have variable romanizations in the resulting training corpus.

Training set unioned with B-A training		Classifier F1 fastText	mT5-base	mT5-large
None (B-A alone)		81.4	84.9	85.3
Dakshina	2-g sampled	88.6	**86.6**	87.7
	3-g sampled	88.7	86.5	**89.0**
	4-g sampled	**88.8**	**86.6**	88.3
Aksharantar	3-g sampled	87.2	84.8	85.7

Table 6: Dakshina development set performance as a function of training set and model class, comparing baseline B-A synthesized training data with other sampled synthesis methods when they are unioned with the baseline training data.

tar romanization lexicons. Here the romanization for each word is the highest probability (1-best) transliteration according to the model.

We can make a couple observations from these results. First, none of the romanization models quite reach the performance achieved with the baseline synthesized corpus, but improvements are achieved with the Dakshina-trained models as the order of the pair n-gram models increases. The Aksharantar 3-gram model resulted in synthesized training sets that did not quite reach the level of Dakshina 3-gram model, despite most Dakshina romanization lexicon entries being included in Aksharantar for all of these languages.[17]

4.2 Sampled romanizations

Table 5 compares systems trained on the baseline training set with those synthesized by sampling from pair n-gram transliteration models of various orders when trained on either Dakshina or Aksharantar romanization lexicons.

In contrast to systems reported in Table 4, these systems achieved large improvements over the baseline, particularly fastText systems. For these methods, the pair n-gram model order made less difference than in Table 4. The Aksharantar trained model again yielded synthesized training data that resulted in somewhat less performant LID systems versus the Dakshina conditions, though that condition, too, improved on the baseline. Interestingly, for training data synthesized in this way, the best fastText system outperformed the best mT5 system.

4.3 Combining synthetic datasets

Another source of variation beyond sampling is to combine independently synthesized training sets. To that end, Table 6 presents conditions within which our newly created synthetic training sets were combined with the baseline synthetic training set, thus yielding twice the amount of text per language – each sentence repeated twice, typically romanized distinctly. All conditions improve from the corresponding systems reported in Table 5, though lower-order pair n-gram conditions achieved larger gains leading to similar performance across conditions. In the interest of conciseness, we focus on Pair 3-gram conditions in future results.

4.4 Sampling multiple copies

If combining two distinctly romanized versions yielded some improvements, then might training on several sampled romanizations yield further gains? Note that, when sampling from the pair n-gram models, each pass over the provided training set will yield distinctly romanized training data. Table 7 compares the baseline and 3-gram results from Table 6 (which used one sampled training set and the B-A baseline training set), with systems trained on an additional 9 distinctly sampled romanized training sets (for a total of 10 plus the baseline). This yielded a modest improvement for the fastText classifier in the Dakshina condition, resulting in our best result (89.2% F1) using purely synthetic training data. None of the mT5 model conditions improved with extra copies, nor did the Aksharantar conditions. See Appendix B for more analysis of varying the quantity of synthetic data.

4.5 Adding harvested data

Table 8 presents the addition of harvested text (see Section 3.4) to earlier reported systems, including the baseline and the best synthesized training data condition. The MADLAD-400 harvested data only degrades the baseline when added to LID training,

[17] We only show Pair 3-gram results for Aksharantar in the interest of conciseness, since that is the order that is eventually selected for both Dakshina and Aksharantar, for reasons that will become apparent as more results are shared.

Training set unioned with B-A training		Classifier F1 fastText	mT5-base	mT5-large
None (B-A alone)		81.4	84.9	85.3
Dakshina	x 1	88.7	**86.5**	**89.0**
3-g sampled	x 10	**89.2**	86.4	88.3
Aksharantar	x 1	87.2	84.8	85.7
3-g sampled	x 10	87.4	84.9	85.9

Table 7: Dakshina development set performance as a function of training set and model class, comparing baseline B-A synthesized training data with other sampled synthesis methods when they are unioned with the baseline training data. Multiple distinctly sampled versions of the training corpus can be created; here we compare the use of a single version with the use of 10 distinct versions.

Training sets unioned with B-A training	Classifier F1 fastText	mT5-base	mT5-large
None (B-A alone)	81.4	84.9	85.3
Dakshina 3g sample x 10	89.2	86.4	88.3
MADLAD noisy	80.3	83.2	84.7
MADLAD clean	80.7	82.4	83.6
GlotCC	86.5	85.4	86.7
GlotCC ⋃ Dak 3-g x 10	**90.5**	**89.2**	**89.6**

Table 8: Dakshina development set performance as a function of training set and model class, comparing baseline B-A synthesized training data with other sampled synthesis methods and harvested data sets when they are unioned with the baseline training data (and each other).

but the GlotCC data is helpful, though not as much on its own as the improved synthetic training sets already presented – perhaps unsurprising given the sparseness of the collection. Combining the best harvested data set (GlotCC) with the data used in the best synthesized condition yields the best observed result by 1.3% absolute F1. See Appendix C for more analysis of varying the quantity of harvested data.

4.6 Test set results

Table 9 presents performance on the full B-A test set of published fastText and pretrained classifier baselines and systems trained for this paper using three training sets: (1) the baseline synthesized training set; (2) the best exclusively synthesized training set; and (3) the best synthesized training set combined with GlotCC harvested data. The observed patterns from the development set hold for this set as well, yielding the best reported results for this task. One notable difference between the dev and test results is that the F1 scores are universally worse than accuracy, indicating that there is some class imbalance in the predictions – unsurprising when some of the languages have an order of magnitude less data in the test set. Still, in absolute terms, the divergence between accuracy and F1 is reduced in the best systems.

4.7 Analysis and discussion

The best synthetic training data (Dakshina 3-gram romanizations sampled 10× plus the baseline B-A synthetic training set) yielded precision, recall and F1 gains for all languages in the development set, as shown in Table 10. Languages which were initially poorly classified, including Hindi, Urdu, Sindhi and Punjabi, show remarkable improvements: 17.9% absolute F1 score improvement for Hindi and 18.0% for Urdu. Figure 1 presents the baseline fastText system's confusion matrix on the development set, as well as a map showing where the differences fell between the baseline and the best synthetic training set. Baseline confusions are evident between (1) Hindi and Urdu; (2) Urdu and Sindhi; and (3) Punjabi and each of Hindi, Sindhi and Urdu. All of these cases improved with the updated synthetic training data. Dravidian languages (Kannada, Malayalam, Tamil, Telugu) had the best baseline performances of any languages, but still managed to achieve positive gains from the updated synthetic training data. See Appendix F for additional qualitative error analysis.

Our hypothesis has been that sampling romanizations mimics natural spelling variation in romanized text, and for that reason using such variations in synthesized training, versus using the same (albeit likely) romanization for each instance of a word, results in better identification of the languages when written in the Latin script. Do the samples actually mimic natural spelling variation? In an effort to answer this question, we extracted frequent alternations resulting from the sampling and coded them according to their correspondence to known alternations cited in the literature. Briefly, the most frequent alternation, accounting for nearly 50% of the instances, corresponded to well-known variations in indication of vowel length or quality—for example, doubling of vowels to indicate length. Another 25% of alternations involved presence/absence of the implicit vowel, and a further 10% the inclusion/omission of 'h' to indicate aspiration or other consonant property. See Appendix D for full details of counting and coding, along with many examples. Overall, this analysis suggests that the romanized samples that we synthesize in this work do indeed mimic spelling

System	fastText Accuracy / F1	Pretrained model Accuracy / F1	model type
IndicLID (Madhani et al., 2023a), published baselines	71.5 / 63.3	80.4 / 74.7	BERT
B-A synthetic training set	80.7 / 71.6	82.4 / 73.1	mT5-large
B-A + Dakshina/Aksharantar hybrid sampled x 10	90.5 / 85.4	89.1 / 83.3	mT5-large
B-A + Dakshina/Aksharantar hybrid sampled x 10 + GlotCC	**92.2 / 88.2**	91.8 / 87.1	mT5-large

Table 9: Accuracy and macro F1 performance on the B-A test set.

Language	Baseline P / R / F1	Best P / R / F1	Diff P / R / F1
Bangla	77.8/ 94.5/ 85.3	89.1/ 96.1/ 92.4	11.3/ 1.6/ 7.1
Gujarati	83.7/ 89.6/ 86.5	89.4/ 93.2/ 91.3	5.8/ 3.6/ 4.8
Hindi	72.2/ 59.9/ 65.5	81.2/ 85.7/ 83.4	9.0/25.8/17.9
Kannada	85.9/ 96.1/ 90.7	91.6/ 96.7/ 94.1	5.7/ 0.6/ 3.4
Malayalam	84.3/ 94.3/ 89.0	87.5/ 94.6/ 90.9	3.2/ 0.3/ 1.9
Marathi	85.5/ 87.1/ 86.3	96.1/ 87.3/ 91.5	10.7/ 0.2/ 5.2
Punjabi	71.1/ 86.5/ 78.0	88.6/ 92.9/ 90.7	17.6/ 6.4/12.7
Sindhi	77.6/ 69.8/ 73.5	86.0/ 86.2/ 86.1	8.5/16.4/12.6
Tamil	93.7/ 93.9/ 93.8	94.4/ 94.9/ 94.7	0.7/ 1.0/ 0.9
Telugu	91.3/ 90.1/ 90.7	92.7/ 91.2/ 91.9	1.4/ 1.1/ 1.2
Urdu	82.1/ 42.8/ 56.3	87.4/ 64.6/ 74.3	5.3/21.8/18.0

Table 10: Dakshina development set per-language precision, recall and F1 when training on (1) the baseline synthetic training set and (2) the best synthetic training set (adding Dakshina 3-gram sampled 10×), as well as the absolute difference (Best minus Baseline). All differences are positive.

variation observed in natural romanizations.

5 Conclusion and Future Work

Through careful experimental analysis on the development set, in this paper we have demonstrated the importance of training text synthesis in improving informally romanized LID. In particular, we show that drawing samples from relatively simple romanization models yields romanizations that capture the lack of Latin orthography and spelling variability in these languages. Independently harvested text was also shown to yield further improvements, though such datasets are sparse for the languages investigated in this paper, so identifying and including more such text constitutes a major future direction. The best system – which is the best reported result for this task by a large margin – is a lightweight linear model, which might further benefit from feature set analysis and augmentation. Additionally, improving LID for informally romanized text outside of South Asia would be of interest, e.g., for Arabic or languages natively written using the Ge'ez script such as Amharic and Tigrinya.

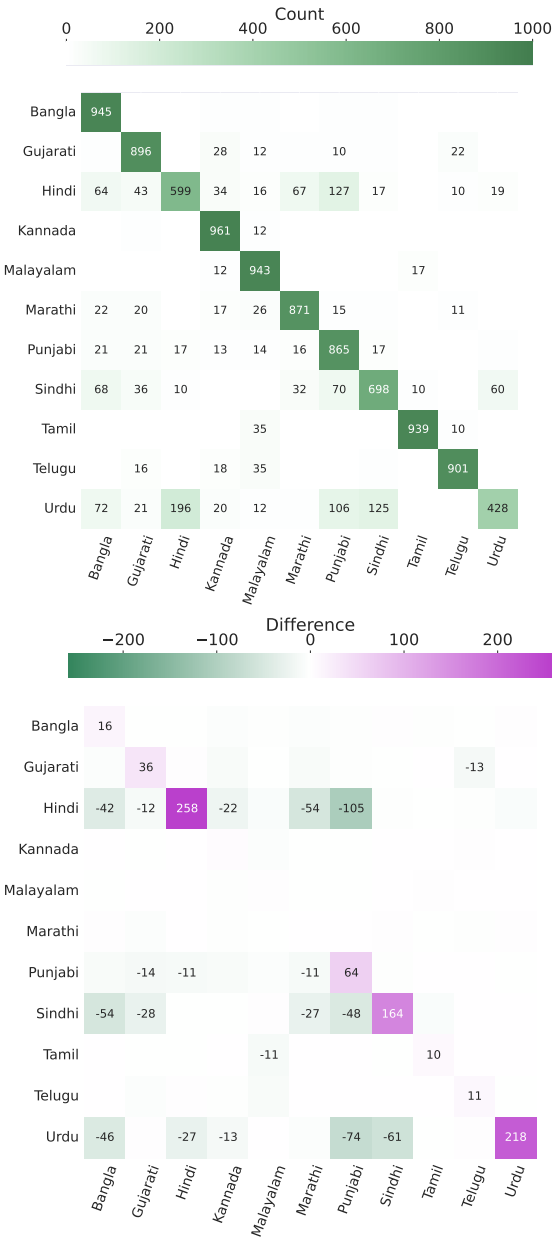

Figure 1: Confusion matrix for a fastText model trained on the baseline B-A synthetic training set (top), and improvement from training on 10× additional samples from a Dakshina-trained pair n-gram transliteration model (bottom). Large, positive values on the diagonal indicate more correctly classified examples, and large, negative values in the off-diagonal entries indicate fewer confusions.

6 Limitations

This work examines a closed-class classification task with a fixed number of labels, which would need to be modified to be applicable to a broader set of languages in real use scenarios. Additionally, the experiments examine performance on a relatively small number of languages (20) from

just three language families (Indo-Aryan, Dravidian and Tibeto-Burman), and hence do not capture the diversity of languages – including, e.g., Semitic languages – for which informal romanization is also common.

The work focuses on dedicated LID systems with a general goal of low computational cost and latency, and does not examine the performance of commercial large language models such as ChatGPT or Gemini.

In this study we investigated LID systems in a scenario where the Latin script is used as an informal common script across various South Asian languages. It has not been established whether approaches that were demonstrated to be effective in this work would yield similar system improvements in scenarios where different scripts (e.g., Perso-Arabic or Devanagari) were being informally used instead of the Latin script.

7 Ethics statement

The goal of this work is to provide methods that advance the field's collective ability to create balanced and inclusive data sets, i.e., that include representative data from typically under-represented languages as well as from common yet chronically under-represented non-standard use scenarios, in addition to well-represented languages and conditions. Such non-standard use scenarios may include writing in informal registers and/or with non-standard scripts or spellings, which are important forms of written communication worldwide.

Acknowledgements

The authors would like to thank Cibu Johny and Raiomond Doctor for their help with this paper.

References

Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. Towards a cleaner document-oriented multilingual crawled corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4344–4355, Marseille, France. European Language Resources Association.

Ife Adebara, AbdelRahim Elmadany, Muhammad Abdul-Mageed, and Alcides Inciarte. 2022. AfroLID: A neural language identification tool for African languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1958–1981, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wafia Adouane, Nasredine Semmar, and Richard Johansson. 2016. Romanized Berber and Romanized Arabic automatic language identification using machine learning. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 53–61, Osaka, Japan. The COLING 2016 Organizing Committee.

Umair Z Ahmed, Kalika Bali, Monojit Choudhury, and Sowmya VB. 2011. Challenges in designing input method editors for Indian languages: The role of word-origin and context. In *Proceedings of the Workshop on Advances in Text Input Methods (WTIM 2011)*, pages 1–9, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Alham Fikri Aji, Genta Indra Winata, Fajri Koto, Samuel Cahyawijaya, Ade Romadhony, Rahmad Mahendra, Kemal Kurniawan, David Moeljadi, Radityo Eko Prasojo, Timothy Baldwin, Jey Han Lau, and Sebastian Ruder. 2022. One country, 700+ languages: NLP challenges for underrepresented languages and dialects in Indonesia. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7226–7249, Dublin, Ireland. Association for Computational Linguistics.

Elay Annamalai and Sanford B. Steever. 2015. Modern Tamil. In *The Dravidian Languages*, 2nd edition, Routledge Language Family Series, chapter 3, pages 118–175. Routledge.

Soubeika Bahri. 2022. Ettounsi and Tamazight writing on Facebook: Oral vernaculars or new literacies. In Cecelia Cutler, May Ahmar, and Soubeika Bahri, editors, *Digital Orality: Vernacular Writing in Online Spaces*, pages 65–93. Springer International Publishing, Cham, Germany.

Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2022. Hierarchical aggregation of dialectal data for Arabic dialect identification. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4586–4596, Marseille, France. European Language Resources Association.

Hemanta Baruah, Sanasam Ranbir Singh, and Priyankoo Sarmah. 2024. Transliteration characteristics in romanized Assamese language social media text and machine transliteration. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 23(2):1–36.

Kenneth R Beesley. 1988. Language identifier: A computer program for automatic natural-language identification of on-line text. In *Proceedings of the 29th annual conference of the American Translators Association*, volume 47, page 54.

Tej Krishan Bhatia. 2010. *Punjabi: A cognitive-descriptive grammar*. Descriptive Grammars. Routledge.

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

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.

William B Cavnar and John M Trenkle. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, volume 161175, page 14, Las Vegas, NV.

Wei-Rui Chen, Ife Adebara, Khai Doan, Qisheng Liao, and Muhammad Abdul-Mageed. 2024. Fumbling in Babel: An investigation into ChatGPT's language identification ability. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4387–4413, Mexico City, Mexico. Association for Computational Linguistics.

Nishaant Choksi. 2020. From transcript to "transcript": Romanized Santali across semiotic media. *Signs and Society*, 8(1):62–92.

Kenneth Church. 1986. Stress assignment in letter to sound rules for speech synthesis. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 2423–2426, Tokyo, Japan. IEEE.

Giovanni Ciotti. 2017. On (the) *sandhi* between the Tamil and Sanskrit grammatical traditions. *Histoire Epistémologie Langage*, 39(2):89–102.

Çağrı Çöltekin, Taraka Rama, and Verena Blaschke. 2018. Tübingen-Oslo team at the VarDial 2018 evaluation campaign: An analysis of n-gram features in language variety identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 55–65, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20:273–297.

Isin Demirsahin, Cibu Johny, Alexander Gutkin, and Brian Roark. 2022. Criteria for useful automatic Romanization in South Asian languages. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6662–6673, Marseille, France. European Language Resources Association.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sayantan Dey, Shivam Thakur, Akhilesh Kandwal, Rohit Kumar, Sharmistha Dasgupta, and Partha Pratim Roy. 2024. BharatBhasaNet - a unified framework to identify Indian code mix languages. *IEEE Access*, 12:68893–68904.

Lars Hellsten, Brian Roark, Prasoon Goyal, Cyril Allauzen, Françoise Beaufays, Tom Ouyang, Michael Riley, and David Rybach. 2017. Transliterated mobile keyboard input via weighted finite-state transducers. In *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing (FSMNLP 2017)*, pages 10–19, Umeå, Sweden. Association for Computational Linguistics.

Ann Irvine, Jonathan Weese, and Chris Callison-Burch. 2012. Processing informal, Romanized Pakistani text messages. In *Proceedings of the Second Workshop on Language in Social Media*, pages 75–78, Montréal, Canada. Association for Computational Linguistics.

Tommi Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2019. Automatic language identification in texts: A survey. *Journal of Artificial Intelligence Research*, 65:675–782.

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*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association*

for Computational Linguistics: Volume 2, Short Papers, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

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. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, Vancouver, Canada.

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.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

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.

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.

Durga Prasad Manukonda and Rohith Gowtham Kodali. 2025. byteSizedLLM@NLU of Devanagari script languages 2025: Language identification using customized attention BiLSTM and XLM-RoBERTa base embeddings. In *Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL 2025)*, pages 248–252, Abu Dhabi, UAE. International Committee on Computational Linguistics.

G. Bharathi Mohan, R. Prasanna Kumar, R. Elakkiya, R. Venkatakrishnan, Shankar Harrieni, Y. Sree Harshitha, K. Harini, and M. Nikhil Reddy. 2023. Transformer-based models for language identification: A comparative study. In *2023 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pages 1–6, Puducherry, India. IEEE.

Seppo Mustonen. 1965. Multiple discriminant analysis in linguistic problems. *Statistical Methods in Linguistics*, 4:37–44.

Elizabeth Nielsen, Christo Kirov, and Brian Roark. 2023. Distinguishing Romanized Hindi from Romanized Urdu. In *Proceedings of the Workshop on Computation and Written Language (CAWL 2023)*, pages 33–42, Toronto, Canada. Association for Computational Linguistics.

Kosuru Pavan, Niket Tandon, and Vasudeva Varma. 2010. Addressing challenges in automatic language identification of romanized text. In *8th International Conference on Natural Language Processing (ICON 2010)*, Kharagpur, India.

Sandun Sameera Perera, Lahiru Prabhath Jayakodi, Deshan Koshala Sumanathilaka, and Isuri Anuradha. 2025. IndoNLP 2025 shared task: Romanized Sinhala to Sinhala reverse transliteration using BERT. In *Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages*, pages 135–140, Abu Dhabi. Association for Computational Linguistics.

Janet Pierrehumbert and Rami Nair. 1996. Implications of Hindi prosodic structure. *Current trends in phonology: Models and methods*, 2:549–584.

Mark W. Post and Robbins Burling. 2017. The Tibeto-Burman languages of Northeast India. In Graham Thurgood and Randy J. LaPolla, editors, *The Sino-Tibetan Languages*, 2nd edition, Language Family Series. Routledge, London.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(1):5485–5551.

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.

Harold F. Schiffman. 1999. *A reference grammar of spoken Tamil*. Reference Grammars. Cambridge University Press, Cambridge, UK.

Mads Toftrup, Søren Asger Sørensen, Manuel R. Ciosici, and Ira Assent. 2021. A reproduction of apple's bi-directional LSTM models for language identification in short strings. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 36–42, Online. Association for Computational Linguistics.

Zubair Torwali. 2020. Countering the challenges of globalization faced by endangered languages of North Pakistan. *Language Documentation and Description*, 17:44–65.

Lawrence Wolf-Sonkin, Vlad Schogol, Brian Roark, and Michael Riley. 2019. Latin script keyboards for South Asian languages with finite-state normalization. In *Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing*, pages 108–117, Dresden, Germany. Association for Computational Linguistics.

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.

A LID Training details

All neural models were finetuned with a constant learning rate of 10^{-3} for 50,000 iterations of batch size 64, with an input sequence length of 256 SentencePiece tokens (Kudo and Richardson, 2018). This matches the finetuning described in Xue et al. (2021), and took 164.3 TensorCore-hours on a Cloud TPU v3[18] for an mT5-large model. FastText models were trained directly on the supervised training set (no unsupervised pretraining), with hidden dimension 16, and all character n-grams in the range [3, 7].

No Punct	Char n-gram order Min	Max	Accuracy	Macro F1
✗	1	3	81.1	80.4
✓	1	3	81.5	80.9
✓	2	4	81.3	80.5
✓	3	4	81.0	80.1
✓	1	5	81.9	81.1
✓	2	5	81.6	80.7
✓	3	5	81.8	80.9
✓	1	6	82.1	81.2
✓	2	6	81.9	81.1
✓	3	6	81.9	81.0
✓	3	7	82.2	81.4
✓	3	8	82.1	81.3
✓	3	9	**82.3**	**81.5**
✓	4	7	82.0	81.3
✓	4	8	82.1	**81.5**
✓	4	9	82.0	81.3

Table 11: fastText Dakshina development set % performance as a function of hyperparameters. Models are trained on the released B-A romanized training set restricted to the Dakshina languages, with hidden layer dimension 16. We selected character n-grams in [3, 7], since we found that setting performed well, with further increase to the min/max character n-gram value yielding marginal performance gain.

Table 11 presents development set accuracy and F1 as these metaparameters were varied. We removed all non-alphanumeric characters as part of our preprocessing, as we found that these features did not generalize well on the development set. Initially we found that the fastText models picked up on punctuation as being indicative of Kashmiri – possibly an artifact of the domain from which the Kashmiri examples were sourced. FastText training is cheap, on the order of minutes purely using CPU.

B Varying the number of synthetic training examples

For all orders of pair n-gram models, synthesizing more sampled training data tended to improve development set performance, however these gains were marginal relative to the gain from training on just a single synthesized version of the training data, e.g., 81.4 → 88.0 F1 vs. 88.0 → 89.0 F1. Figure 2 shows how F1 varies as the number of synthetic copies of the training data is increased for a range of models.

[18] https://cloud.google.com/tpu/docs/v3

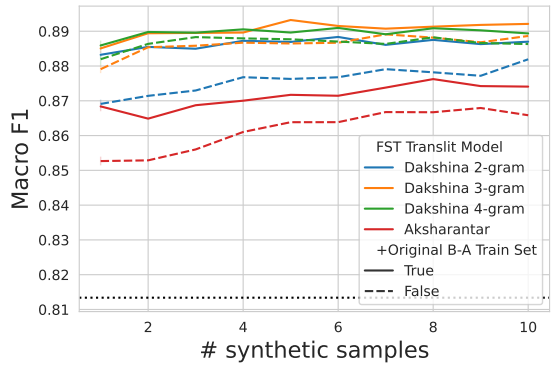

Figure 2: Macro F1 for various synthetic training sets as a function of number of samples to train a fastText LID model on. Solid lines indicate that the LID model was also trained on the original B-A data, while a dotted line indicates only training on synthesized samples from the pair n-gram model. Baseline performance is indicated by the dotted black line at the bottom.

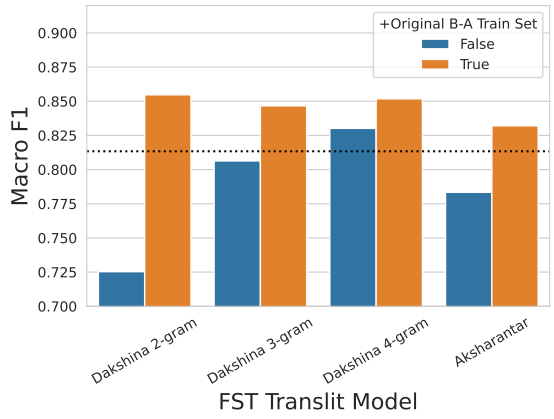

Figure 3: Macro F1 for various synthetic training sets, when decoding the 1-best candidate for each native word under the pair n-gram transliteration model, rather than sampling from the 8-best candidate list. The dotted black line indicates baseline performance of training on the original synthetic training set.

While LID models trained on Dakshina pair n-gram model derived training data tend to perform well, irrespective of the order of the pair n-gram model, the samples from the Aksharantar-trained pair n-gram models are strictly worse. This gap persists even when also training on the original B-A training set. While the 1-best candidate generated from these pair n-gram models are complementary to the released training set, by themselves, the 1-best candidate can be quite poor (Figure 3). For example, training only on samples from the 2-gram pair LM yields 72.5 F1, far worse than the baseline of 81.4. But combining that data with the baseline training set yields strong improvements.

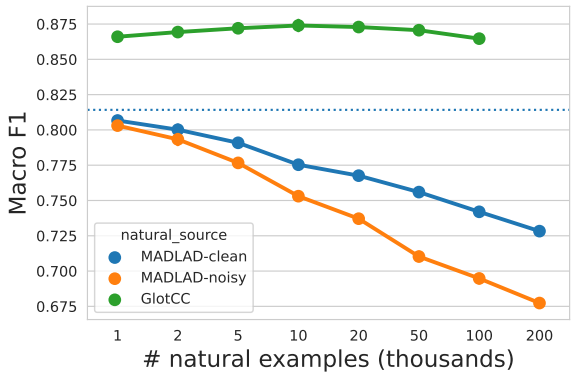

Figure 4: Macro development set F1 as a function of training corpus and number of examples added to each class. The original B-A training set is included in all runs. The dashed line corresponds to the performance of training only on the original B-A synthetic training set.

C Varying the number of harvested training examples

Figure 4 presents development set F1 as the number of sampled examples added to the B-A training set is varied in $\{1, 2, 5, 10, 20, 50, 100, 200\}$ thousand examples. A fastText model was retrained in each case, and evaluated on the development set. We replicated each of these runs 5 times, averaging performance over the runs. Note that the original training set contains 100k examples per class, so adding up to 1k examples per class resulted in at most a 1% increase in the training set size.

While the addition of more MADLAD data actively hurt the performance, the addition of the "noisy" subset harms the LID model more than the "clean" subset as the number of added examples increases. GlotCC is shown to be useful, even though this data also contains noisy labels, and does not cover all of the Dakshina classes. See Table 15 for some illustrative examples from the GlotCC dataset. Label quality is paramount in training romanized LID models. Harvested text has little value, or can even be actively harmful, if it is unlikely to actually be the language of interest.

D Spelling variation in synthetic romanized samples

We find that training on synthetic samples of romanized text improves LID classification performance. Do these synthetic samples actually reflect natural spelling variation? In this section, we compare the variation resulting from sampling versus 1-best decoding from the pair 3-gram transliteration models trained on Dakshina romanization lexicons. Because each decoded/sampled token

was derived from the same native script string, we are able to directly compare spelling variation between romanized words generated by each method, token-by-token.

Over all Dakshina languages, 31% of synthetically romanized tokens differ between the 1-best decoded and sampled romanizations. To find common patterns of these differences, we collected counts of character 5-gram minimum Levenshtein distance edits between the 1-best decode vs. sampled tokens within each language's synthetic training set. For example, we counted all instances where a substring "arne" in the 1-best romanization was sampled as "arane" by the FST romanization model, or "attu" was sampled as "atthu". From these counts we can identify common surface edit patterns.

How nasals, implicit vowels, aspiration, vowel quality, and voicing are rendered in romanizations can vary in a language – sometimes these phenomena are clearly indicated in the Latin script, sometimes not (Roark et al., 2020; Demirsahin et al., 2022; Kirov et al., 2024). We coded each of the 20 most frequent 5-gram edit patterns within a language by surface pattern type (Table 12). Each of these patterns are representative of natural romanization variation attested in the literature and we share the full set of annotated patterns in Table 13.

Of these surface patterns, inserting vowels next to an existing one to indicate quality or length (label **L**, 49%) was the most common. This includes alternation between "u" ↔ "oo" and "i" ↔ "ee" to indicate IPA /u/, /i/ respectively, along with doubling of vowels to explicitly indicate vowel quality (e.g., to distinguish from *schwa*), "a" ↔ "aa", "e" ↔ "ee", and "i" ↔ "iy".

The second most common pattern (label **I**, 25.5% of patterns) was addition/deletion of a lone vowel following a consonant. This is indicative of whether the implicit vowel (e.g., schwa) was explicitly written in the romanization. For the surface patterns we see here, this vowel is "a" (e.g., in Hindi "yojan" → "yojn").

The addition of an "h" following a consonant often indicates aspiration (label **H**, 9.5%), but may also clarify some other property of the consonant. For instance, in Tamil, "th" and "dh" respectively indicate unvoiced and voiced *dental* stops – the voicing property depends on the context and is not indicated in native Tamil orthography (Schiffman, 1999; Annamalai and Steever, 2015). Both of these correspond to unaspirated consonants, where the presence of "h" instead distinguishes them from their retroflex counterparts.

The final most frequent pattern was doubling of a consonant at the same place of articulation, e.g., "tha" → "ththa" (label **G**, 7% of patterns). This pattern occurred frequently in Tamil and Punjabi edit strings. In Tamil, a Dravidian language with agglutinative morphology, gemination is often due to the *sandhi* effect, a set of phonological changes occurring at the location where morphemes combine (Ciotti, 2017). Gemination is typical for Punjabi as well, unlike other Indo-Aryan languages (Bhatia, 2010).

Other patterns occurring in less than 5% of edits were alternation in the choice of a back/mid vowel (e.g., frequent a/o alternation in Bangla), changing the voicing of a consonant perhaps due to coarticulation effects with a neighboring vowel (e.g., "aigal" → "aikal"), and the addition/deletion of a syllable-final nasal due to transliterating the *anusvara* character explicitly ("nahin" → "nahi").

Out of these 200 frequent 5-gram edit patterns, we were only unable to classify three instances as one of the given classes. Of these, "dwara" → "dvara" in Hindi indicates a plausible variation, as v/w are allophones of each other in Hindustani (Pierrehumbert and Nair, 1996). In Marathi, "madhy" → "madh" and "adhye" → "adhe" indicates Latin spelling variation in a common morpheme, मध्ये, meaning "in" or "amid".

E Evaluation on a synthetic development set

We chose to develop LID models on natural romanized examples from the Dakshina development set, restricting ourselves to the subset of Dakshina languages where we had natural examples to evaluate on. Table 14 displays the performance of these models on the synthetic development set for the same subset of languages.

We find that model performance is higher than what we found in Tables 5 to 8. This agrees with the observation in Table 5 of Madhani et al. (2023a) that LID performance on automatically transliterated data is clearly inflated over naturally-generated text. The variation in performance across different training sets is also narrowed, potentially making training set selection more difficult: between 82.2% and 90.5% accuracy for the natural development set vs. 87.2% to 93.1% for the synthetic.

Label	Count (%)	Description
A	9 (4.5)	Change a low/mid vowel (**A**/e/o). Indicative of variation in pronunciation.
G	14 (7.0)	Inserting an additional consonant at the same place of articulation. Indicative of **G**emination.
H	19 (9.5)	Addition of **H** following a consonant. Indicative of aspiration.
I	51 (25.5)	Inserting/deleting a vowel after a consonant. Indicative of **I**mplicit vowel representation.
L	98 (49.0)	Addition/deletion of a vowel next to existing vowel. Indicative of vowel **L**ength/quality.
N	2 (1.0)	Addition of syllable-final **N**asal. Indicative of variation in representing *anusvara*.
V	5 (2.5)	Consonant change at same place of articulation. Indicative of alternation in **V**oicing

Table 12: Description of spelling variation labels in Table 13. Counts are over a total of 200 5-gram edit patterns, the 20 most frequent per language. Counts do not sum to 200 as one edit pattern was labeled as both **A** and **L**, and three patterns fell outside of this coding scheme.

F Error analysis

Comparing individual development set predictions from the baseline fastText model trained on B-A synthetic training data with the model trained on the best performing synthetic training data (B-A training data combined with Dakshina 3-gram sampled x 10 training data), we find that out of the 11,000 examples in the development set: (1) 8,847 examples were correctly classified by both models; (2) 163 were regressions from the baseline; (3) 989 were newly classified correctly by the updated model; and (4) 1,001 examples were incorrectly classified by both models. It is worth noting that the examples that were incorrectly classified by the updated model are markedly shorter than the ones which were correctly classified – median of 3 tokens and 20 characters for the "both lose" case, 12 tokens and 90 characters for the "both win", 4 tokens and 29 characters for the "regression" case, and 11 tokens and 64 characters for the "corrected" case. Table 16 includes some examples randomly selected from each group. Many of these incorrect examples contain English strings, either wholly or in part,[19] and proper names are also common (possibly confusing the LID model).

[19]Note that the B-A benchmark removed many such items from their test set, so this is one difference between our development and test sets.

Cnt x 1000	1-best n-gram	Sample n-gram	La-bel	Cnt x 1000	1-best n-gram	Sample n-gram	La-bel	Cnt x 1000	1-best n-gram	Sample n-gram	La-bel	Cnt x 1000	1-best n-gram	Sample n-gram	La-bel
	Bangla				**Gujarati**				**Hindi**				**Kannada**		
4.0	yeech	yech	L	3.4	maate	mate	L	2.4	karn	karan	I	3.2	avag	avaag	L
3.6	eeche	eche	L	3.2	karv	karav	I	2.1	kart	karat	I	2.6	vagi	vaagi	L
3.3	ebong	abong	A	2.8	vama	vaman	M	1.7	arne	arane	I	2.3	hara	haara	L
3.1	hayee	haye	L	2.8	kari	karee	L	1.6	rajya	rajy	I	2.2	kara	kaara	L
2.5	ayeec	ayec	L	2.3	arva	arava	I	1.5	yojan	yojn	I	2.0	havaa	hava	L
2.4	hayee	hoye	AL	2.2	yare	yaare	L	1.5	kuch	kuchh	H	1.9	thava	thav	I
2.0	ebong	ebang	A	2.1	athi	athee	L	1.4	ojana	ojna	I	1.9	alag	alaag	L
1.9	samy	samay	I	1.9	hati	hatee	L	1.3	hetr	hetra	I	1.8	matt	matth	H
1.7	koren	karen	A	1.8	arva	arvaa	L	1.2	sakt	sakat	I	1.7	agid	aagid	L
1.6	kore	korey	L	1.8	aman	amaan	L	1.2	dwara	dvara	!W	1.5	akar	akaar	L
1.6	heke	hekey	L	1.8	rite	reete	L	1.2	lakin	lekin	A	1.5	attu	atthu	H
1.6	bosth	basth	A	1.8	thay	thaay	L	1.1	bhara	bhar	I	1.4	tara	thara	H
1.6	ayeec	oyec	L	1.8	hata	hataa	L	1.1	harat	hart	I	1.4	iyag	iyaag	L
1.5	eche	echey	L	1.7	dhar	dhaar	L	1.1	karya	kary	I	1.3	haagu	hagu	L
1.5	proti	prati	A	1.6	hava	havaa	L	1.1	arte	arate	I	1.3	aman	amaan	L
1.5	chil	chhil	H	1.6	vama	avama	I	1.1	arti	arati	I	1.3	aagu	aagoo	L
1.5	korec	karec	A	1.6	aara	aaraa	L	1.1	nahin	nahi	M	1.3	haag	haago	I
1.4	eche	echhe	H	1.6	vama	vamaa	L	1.0	karan	karn	I	1.3	anta	antha	H
1.4	tini	teeni	L	1.5	adha	adhaa	L	1.0	tarh	tarah	I	1.3	mana	maana	L
1.4	korte	karte	A	1.5	tyar	tyaar	L	0.9	pahl	pahal	I	1.3	kari	kaari	L
	Malayalam				**Marathi**				**Punjabi**				**Tamil**		
3.8	nathu	nath	H	4.0	karn	karan	I	9.1	icch	ichch	H	8.4	athth	ath	G
3.5	amaay	amay	L	2.4	arny	arany	I	9.1	vicc	vichc	H	7.4	ththi	thi	G
2.4	yaanu	yanu	L	2.4	rnya	ranya	I	6.4	vicch	vich	G	5.9	nth	nthth	G
2.4	thaan	than	L	2.3	arata	arta	I	3.3	karan	karn	I	5.4	ththu	thu	G
2.3	maayi	mayi	L	1.8	athi	athee	L	2.3	dian	diaan	L	5.3	argal	arkal	V
2.1	haana	hana	L	1.7	karat	kart	I	2.0	keeta	kita	L	5.3	uthth	uth	G
1.9	maaya	maya	L	1.7	asun	asoon	L	2.0	dian	diyan	L	5.1	ththa	tha	G
1.8	aayir	ayir	L	1.5	achi	achee	L	1.9	jand	jaand	L	4.6	ithth	ith	G
1.8	ikka	ikkaa	L	1.5	harat	hart	I	1.6	khia	khiaa	L	4.4	antha	andha	V
1.8	athaa	atha	L	1.4	arna	arana	I	1.5	ahin	aheen	L	4.2	hthil	thil	G
1.8	sthaa	stha	L	1.4	sath	sathe	I	1.4	hian	hiaan	L	4.1	aigal	aikal	V
1.7	kkan	kkaan	L	1.3	adhi	adhee	L	1.3	keeti	kiti	L	4.1	ththa	ttha	H
1.6	amaan	aman	L	1.3	bhara	bhar	I	1.3	anda	aanda	L	4.0	anga	angka	G
1.5	aanam	anam	L	1.3	madhy	madh	!Y	1.2	nahi	nahee	L	3.8	runth	rundh	V
1.4	hamaa	hama	L	1.2	adhye	adhe	!Y	1.2	bach	bacch	G	3.7	ngal	ngkal	G
1.4	maanu	manu	L	1.2	arun	aroon	L	1.0	aria	ariaa	L	3.7	ithth	itth	H
1.3	undaa	unda	L	1.1	nyat	anyat	I	1.0	ghat	ghatt	G	3.5	tha	ththa	G
1.3	laanu	lanu	L	1.1	mhana	mhan	I	1.0	karan	kran	I	3.5	ththu	tthu	H
1.3	thram	tram	H	1.1	arne	arane	I	1.0	vale	vaale	L	3.4	athth	atth	H
1.3	ayaan	ayan	L	1.1	hoti	hotee	L	0.9	arti	arati	I	3.2	galai	kalai	V
	Telugu				**Urdu**										
4.2	unna	unnaa	L	16.0	ahein	ahin	L								
3.0	nchaa	ncha	L	14.5	nahei	nahi	L								
2.8	incha	inch	I	8.6	allah	alla	H								
2.7	nnar	nnaar	L	8.1	ahein	ahen	L								
2.7	chaar	char	L	7.4	nahei	nahe	L								
2.5	amlo	amloo	L	6.2	sath	saath	L								
2.3	naru	naaru	L	5.8	stan	astan	I								
2.2	hara	haara	L	5.3	ksta	kasta	I								
2.1	anta	antha	H	5.2	akst	akast	I								
2.0	aalan	alan	L	5.2	paks	pakas	I								
1.9	tunn	tunna	I	4.4	rahe	rahay	L								
1.9	aanik	anik	L	4.1	waqat	waqt	I								
1.8	haaru	haru	L	4.1	khla	khala	I								
1.7	anik	aanik	L	3.8	stan	istan	I								
1.7	chaal	chal	L	3.7	arne	arnay	L								
1.7	aanni	anni	L	3.6	ksta	kista	I								
1.6	tara	thara	H	3.6	akst	akist	I								
1.6	nnay	nnaay	L	3.6	paks	pakis	I								
1.6	tana	thana	H	3.5	karn	karna	I								
1.6	dhaan	dhan	L	3.3	kart	karta	I								

Table 13: The twenty most frequent 5-gram spelling variations between 1-best and sampled romanized tokens per Dakshina language. Each instance is labeled according to the key in Table 12. The three exceptions noted in the text are denoted by "!W" and "!Y" labels. Counts (in thousands, *Cnt x 1000*) are over the entire B-A training set within each language.

Included training data	Not Including B-A		Including B-A	
	fastText Acc / F1	mT5-large Acc / F1	fastText Acc / F1	mT5-large Acc / F1
None			87.2 / 86.8	89.3 / 89.2
Dakshina 3-gram 1x samples	90.5 / 90.3	89.4 / 89.3	92.1 / 92.0	91.8 / 91.7
Dakshina 3-gram 10x samples	**91.4 / 91.3**	89.7 / 89.7	92.4 / 92.3	91.4 / 91.3
Dakshina 2-gram	88.8 / 88.6	88.8 / 88.7	91.8 / 91.6	90.9 / 90.8
Dakshina 4-gram	91.1 / 90.9	**90.0 / 90.0**	92.0 / 91.9	91.2 / 91.2
Aksharantar 3-gram	89.2 / 88.8	88.8 / 88.4	90.9 / 90.7	89.8 / 89.5
Aksharantar 3-gram 10x samples	90.5 / 90.3	89.3 / 89.0	91.4 / 91.2	89.9 / 89.6
GlotCC			90.7 / 90.6	91.4 / 83.8
Dakshina 3-gram 10x + GlotCC			**93.1 / 93.2**	**92.4 / 92.4**

Table 14: Model performance on the synthetic development set released with the B-A corpus, for languages included in Dakshina.

Language	Examples
Bangla	“ALLAH AMR ROB, NOBI AMR SOB. ISLAM AMR DHORMO, NAMAZ UTTOM KORMO.”, “aar aami dekhlam, maar chokh duto anande nachchhe.”, “usher alter”, “This story was co-written by a member of our community using our AI powered storyteller.”
Gujarati	“Happy Holi quotes and status”, “Himalaya Rudraksh & Gems Testing Lab - India’s Most Trusted Rudraksha, Diamond & Gemstone Testing Laboratory”, “Rasulullah Syed al Mursalin ane Khaatam al Nabiyyin chhe.”
Hindi	, “Shamooael 30:1 teesare din jab daud apane janon samet sikalag pahuncha, tab unhon ne kya dekha, ki amalekiyon ne daakkhian desh aur sikalag par chaddhai kee. aur sikalag ko mar ke foonk diya, 6.”, “Natural Ways to Improve Memory in Hindi: Yaaddasht Badhaye”, “Cricket satta ka vikas ek aise vyavsay ki or ishara karta hai jo samay ke saath badalta hai.”
Sindhi	“room aeron chari room aeron chai rroom aeron chairr oom aeron chair orom aeron chair room aeron chair romo qeronchairroom weronchairroom seronchairroom xeronchairroom zeronchairroom eeronchairroom aaeronchairroom eronchairroom a2ronchairroom a3ronchairroom a4ronchairroom awronchairroom arronchairroom asronchairroom adronchairroom afronchairroom aaronchairroom aeeronchairroom...”, “If you like to book room in a Aeron chair room use ““Check price and availability or” ““Book now”” green button, then you will be redirected to the main booking site from our partners, where you would select date of booking and check prices and availability of hotel rooms.”, “Superstar Ayeza Khan touches new skies of popularity by performing in hit dramas “Chupke Chupke” and “Mere Pass Tum Ho”.”
Tamil	“appoathu moayeesan avarka’lai noakki: kadavu’lin aaseer ungka’lukkuk kidaikkumpadi in’ru ungka’lil ovvoruvanum than than makanaiyum sakoatharanaiyum pazhivaangkinamaiyaal, aa’ndavarukku ungka’l kaika’lai arppa’nam seytheerka’l en’raar.”, “paaravoanum avan oozhiyarka’l anaivarum ekipthiyar yaavarum iravil ezhunthanar.”, “Naan kankalai mooti thoonguvadhu pol natiththen.”
Telugu	“Ela undi ani adiga chala bagundi eppudu ela enjoy cheyaledu ani hug chesukundi night 12 ayyindi elago evaru leru chuttu koncham rest tesukoni tent pakkana plana chesam pakkana fire undi”, “Prati samvatsaram January 1 na Global family Day jarupukuntaru .”, “Ee Nagaraniki Emaindi Meme Movie: We Arranged The Entire Movie In Meme Templates.”
Urdu	“Koi khuwahish nahi is deewanay ki”, “Tamaam hamd us Allah ke liye hai jo chupi hui cheezoun ki gehraaioun mein utra hua hai.”, “Click here `https://youtu.be/iLyCJGOU7Js?si=jHKBes_6T9DjYw0s`”

Table 15: Romanized sentences from a selection of development set languages in the GlotCC corpus. While some strings are plausibly the correct language, some English strings and boilerplate are included.

Bucket	Gold Label	Baseline System	Updated (Best)	Example	Number of Tokens	Chars
both win	Telugu	Telugu	Telugu	bhakti paaravasyamu saranaagati ivi ee aaluvaarula jeevitamlonoo rachanalalonoo vaarini gurinchina gaathalalonoo pramukhangaa kaanavachche amsaalu	14	146
both win	Punjabi	Punjabi	Punjabi	buneyadi adhikar manukhi azadi da muddla sidhant han ate harek bharti di shakhsiyat de sahi vikaas layi eh jaruri han	20	117
both win	Telugu	Telugu	Telugu	dharinchagaligina computer	2	26
both win	Bangla	Bangla	Bangla	bivinno arthonoitik totto o mukto bazar sunirdishto boishisto aaboshoyk udahoronswarup ekti nikhut bazar shathe nirbhul tottho ebong nikhut protijogitaar	19	153
both win	Bangla	Bangla	Bangla	gobar goho	2	10
both win	Punjabi	Punjabi	Punjabi	parkash kol Unni R ki kahani nu film layi lain di ek surantar yojna vi si	16	73
both win	Malayalam	Malayalam	Malayalam	ennaal netveyarinethire oru velluviliyuyarthaan ithinaayilla	5	60
both win	Tamil	Tamil	Tamil	Jamui makkalavaith thoguthi Inthiya makkalavaikkaana thoguthiyaagum Ithu Biharin 40 makkalavaith thoguthigalil ondru	12	116
both win	Hindi	Hindi	Hindi	2007 Uttar Pradesh vidhan sabha chunav men inhone Uttar Pradesh kr Merath jile ke Merath kaint vidhan sabha nirvachan shetr se Bhajpa ki aur se chunaav men bhag liya	29	165
both win	Punjabi	Punjabi	Punjabi	Paraguay vich hundi gair kanunni jungal vaadhi	7	46
regression	Telugu	Telugu	Kannada	paurushamme pongeraa	2	20
regression	Telugu	Telugu	Kannada	janaganamana	1	12
regression	Urdu	Urdu	Hindi	jinhein ham dekh kar jeete the Nasir	7	36
regression	Marathi	Marathi	Punjabi	Bahut din nacha bhetalo saubhadra	5	33
regression	Bangla	Bangla	Malayalam	Fellow of the Association for Computing Machinary 1994	8	54
regression	Marathi	Marathi	Gujarati	Narahar kurundakar smruti sahity sammelan Nanded	6	48
regression	Urdu	Urdu	Hindi	1819 mein venezuela aur granada ne mil kar jamhooriya banayi jis ka naam columbia rakha gaya	16	92
regression	Urdu	Urdu	Hindi	chataanein aur romaan	3	21
regression	Punjabi	Punjabi	Kannada	es da vikaas pracina russi bhasa valon hoeya	8	44
regression	Telugu	Telugu	Tamil	adi vasanta kaalam	3	18
corrected	Hindi	Bangla	Hindi	wo Royal socity of London ke nirvachit sadasy the	9	49
corrected	Hindi	Punjabi	Hindi	Unhe Lhasa se beijing tak jana tha lekin aisa sambhav nahi hone par Sangpo yani Brahmaputra ya Bhutan ke raste Bharat ane ke nidesh diye gaye the	27	145
corrected	Punjabi	Urdu	Punjabi	Ek jahaad pyar de layi	5	22
corrected	Hindi	Punjabi	Hindi	sohan rahi ke anusar geet vidha sahitya ki sabse kathin evam shresth vidha hai	14	78
corrected	Telugu	Gujarati	Telugu	aasale adiyaasalai nadi vesavi bratukaayene	5	43
corrected	Urdu	Punjabi	Urdu	qaumi parast rahnuma	3	20
corrected	Tamil	Malayalam	Tamil	Aatkalam Kanitham	2	17
corrected	Sindhi	Bangla	Sindhi	saawanu men ute jaa bhagea panhinjon menhon dunad gion hdrie pke te kadhi indaa ahin jite paani kona hondo ahe	20	110
corrected	Hindi	Urdu	Hindi	sardiyon mein yaha bhaari barfbaari hoti hai aur jheel bhi jam jaati hai	13	72
corrected	Tamil	Hindi	Tamil	paarampariya nel	2	16
both lose	Hindi	Malayalam	Sindhi	Cardinal	1	8
both lose	Hindi	Punjabi	Kannada	yogita bali	2	11
both lose	Gujarati	Malayalam	Telugu	ravishankar mahaaraaj	2	21
both lose	Hindi	Marathi	Marathi	Rachana Parulkar Ajebade Panwar	4	31
both lose	Urdu	Kannada	Kannada	kaala shahzada	2	14
both lose	Sindhi	Punjabi	Punjabi	sancho khalifa rashdeen 2 sancho wazir sahibha sancho khalifa rashideen sancho sahiba	12	85
both lose	Malayalam	Kannada	Kannada	narabali	1	8
both lose	Gujarati	Sindhi	Sindhi	tal	1	3
both lose	Urdu	Punjabi	Hindi	agar silsila bharfaj hota ho to silsile ki infiradi istilahaat laziman sifar ki taraf pahunchen gin	16	99
both lose	Tamil	Malayalam	Malayalam	saara thattil	2	13

Table 16: Sample of instances where the baseline/updated (best) synthetically trained models agree/differ.