Comparative Error Analysis in Neural and Finite-state Models for Unsupervised Character-level Transduction

Maria Ryskina[1] **Eduard Hovy**[1] **Taylor Berg-Kirkpatrick**[2] **Matthew R. Gormley**[3]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Computer Science and Engineering, University of California, San Diego
[3]Machine Learning Department, Carnegie Mellon University
mryskina@cs.cmu.edu hovy@cmu.edu
tberg@eng.ucsd.edu mgormley@cs.cmu.edu

Abstract

Traditionally, character-level transduction problems have been solved with finite-state models designed to encode structural and linguistic knowledge of the underlying process, whereas recent approaches rely on the power and flexibility of sequence-to-sequence models with attention. Focusing on the less explored unsupervised learning scenario, we compare the two model classes side by side and find that they tend to make different types of errors even when achieving comparable performance. We analyze the distributions of different error classes using two unsupervised tasks as testbeds: converting informally romanized text into the native script of its language (for Russian, Arabic, and Kannada) and translating between a pair of closely related languages (Serbian and Bosnian). Finally, we investigate how combining finite-state and sequence-to-sequence models at decoding time affects the output quantitatively and qualitatively.[1]

1 Introduction and prior work

Many natural language sequence transduction tasks, such as transliteration or grapheme-to-phoneme conversion, call for a character-level parameterization that reflects the linguistic knowledge of the underlying generative process. Character-level transduction approaches have even been shown to perform well for tasks that are not entirely character-level in nature, such as translating between related languages (Pourdamghani and Knight, 2017).

Weighted finite-state transducers (WFSTs) have traditionally been used for such character-level tasks (Knight and Graehl, 1998; Knight et al., 2006). Their structured formalization makes it easier to encode additional constraints, imposed either by the underlying linguistic process (e.g. monotonic character alignment) or by the probabilistic generative model (Markov assumption; Eisner, 2002). Their interpretability also facilitates the introduction of useful inductive bias, which is crucial for unsupervised training (Ravi and Knight, 2009; Ryskina et al., 2020).

Unsupervised neural sequence-to-sequence (seq2seq) architectures have also shown impressive performance on tasks like machine translation (Lample et al., 2018) and style transfer (Yang et al., 2018; He et al., 2020). These models are substantially more powerful than WFSTs, and they successfully learn the underlying patterns from

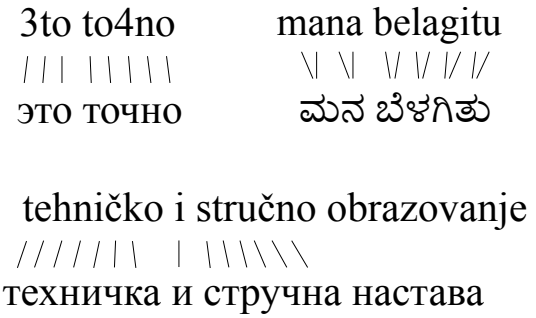

Figure 1: Parallel examples from our test sets for two character-level transduction tasks: converting informally romanized text to its original script (top; examples in Russian and Kannada) and translating between closely related languages (bottom; Bosnian–Serbian). Informal romanization is idiosyncratic and relies on both visual (ч → 4) and phonetic (т → t) character similarity, while translation is more standardized but not fully character-level due to grammatical and lexical differences ('настава' → 'obrazovanje') between the languages. The lines show character alignment between the source and target side where possible.

[1]Code will be published at https://github.com/ryskina/error-analysis-sigmorphon2021

monolingual data without any explicit information about the underlying generative process.

As the strengths of the two model classes differ, so do their weaknesses: the WFSTs and the seq2seq models are prone to different kinds of errors. On a higher level, it is explained by the structure–power trade-off: while the seq2seq models are better at recovering long-range dependencies and their outputs look less noisy, they also tend to insert and delete words arbitrarily because their alignments are unconstrained. We attribute the errors to the following aspects of the trade-off:

Language modeling capacity: the statistical character-level n-gram language models (LMs) utilized by finite-state approaches are much weaker than the RNN language models with unlimited left context. While a word-level LM can improve the performance of a WFST, it would also restrict the model's ability to handle out-of-vocabulary words.

Controllability of learning: more structured models allow us to ensure that the model does not attempt to learn patterns orthogonal to the underlying process. For example, domain imbalance between the monolingual corpora can cause the seq2seq models to exhibit unwanted style transfer effects like inserting frequent target side words arbitrarily.

Search procedure: WFSTs make it easy to perform exact maximum likelihood decoding via shortest-distance algorithm (Mohri, 2009). For the neural models trained using conventional methods, decoding strategies that optimize for the output likelihood (e.g. beam search with a large beam size) have been shown to be susceptible to favoring empty outputs (Stahlberg and Byrne, 2019) and generating repetitions (Holtzman et al., 2020).

Prior work on leveraging the strength of the two approaches proposes complex joint parameterizations, such as neural weighting of WFST arcs or paths (Rastogi et al., 2016; Lin et al., 2019) or encoding alignment constraints into the attention layer of seq2seq models (Aharoni and Goldberg, 2017; Wu et al., 2018; Wu and Cotterell, 2019; Makarov et al., 2017). We study whether performance can be improved with simpler decoding-time model combinations, reranking and product of experts, which have been used effectively for other model classes (Charniak and Johnson, 2005; Hieber and Riezler, 2015), evaluating on two unsupervised tasks: decipherment of informal romanization (Ryskina et al., 2020) and related language translation (Pourdamghani and Knight, 2017).

While there has been much error analysis for the WFST and seq2seq approaches separately, it largely focuses on the more common supervised case. We perform detailed side-by-side error analysis to draw high-level comparisons between finite-state and seq2seq models and investigate if the intuitions from prior work would transfer to the unsupervised transduction scenario.

2 Tasks

We compare the errors made by the finite-state and the seq2seq approaches by analyzing their performance on two unsupervised character-level transduction tasks: translating between closely related languages written in different alphabets and converting informally romanized text into its native script. Both tasks are illustrated in Figure 1.

2.1 Informal romanization

Informal romanization is an idiosyncratic transformation that renders a non-Latin-script language in Latin alphabet, extensively used online by speakers of Arabic (Darwish, 2014), Russian (Paulsen, 2014), and many Indic languages (Sowmya et al., 2010). Figure 1 shows examples of romanized Russian (top left) and Kannada (top right) sentences along with their "canonicalized" representations in Cyrillic and Kannada scripts respectively. Unlike official romanization systems such as pinyin, this type of transliteration is not standardized: character substitution choices vary between users and are based on the specific user's perception of how similar characters in different scripts are. Although the substitutions are primarily phonetic (e.g. Russian н /n/ → n), i.e. based on the pronunciation of a specific character in or out of context, users might also rely on visual similarity between glyphs (e.g. Russian ч /t͡ʃʲ/ → 4), especially when the associated phoneme cannot be easily mapped to a Latin-script grapheme (e.g. Arabic ع /ʕ/ → 3). To capture this variation, we view the task of decoding informal romanization as a many-to-many character-level decipherment problem.

The difficulty of deciphering romanization also depends on the type of the writing system the language traditionally uses. In alphabetic scripts, where grapheme-to-phoneme correspondence is mostly one-to-one, there tends to be a one-to-one monotonic alignment between characters in the ro-

manized and native script sequences (Figure 1, top left). *Abjads* and *abugidas*, where graphemes correspond to consonants or consonant-vowel syllables, increasingly use many-to-one alignment in their romanization (Figure 1, top right), which makes learning the latent alignments, and therefore decoding, more challenging. In this work, we experiment with three languages spanning over three major types of writing systems—Russian (alphabetic), Arabic (abjad), and Kannada (abugida)—and compare how well-suited character-level models are for learning these varying alignment patterns.

2.2 Related language translation

As shown by Pourdamghani and Knight (2017) and Hauer et al. (2014), character-level models can be used effectively to translate between languages that are closely enough related to have only small lexical and grammatical differences, such as Serbian and Bosnian (Ljubešić and Klubička, 2014). We focus on this specific language pair and tie the languages to specific orthographies (Cyrillic for Serbian and Latin for Bosnian), approaching the task as an unsupervised orthography conversion problem. However, the transliteration framing of the translation problem is inherently limited since the task is not truly character-level in nature, as shown by the alignment lines in Figure 1 (bottom). Even the most accurate transliteration model will not be able to capture non-cognate word translations (Serbian 'настава' [nastava, 'education, teaching'] → Bosnian 'obrazovanje' ['education']) and the resulting discrepancies in morphological inflection (Serbian -a endings in adjectives agreeing with feminine 'настава' map to Bosnian -o representing agreement with neuter 'obrazovanje').

One major difference with the informal romanization task is the lack of the idiosyncratic orthography: the word spellings are now consistent across the data. However, since the character-level approach does not fully reflect the nature of the transformation, the model will still have to learn a many-to-many cipher with highly context-dependent character substitutions.

3 Data

Table 1 details the statistics of the splits used for all languages and tasks. Below we describe each dataset in detail, explaining the differences in data split sizes between languages. Additional preprocessing steps applied to all datasets are described in §3.4.[2]

3.1 Informal romanization

Source:	de el menu:)
Filtered:	de el menu<...>
Target:	<...>دي ألنه
Gloss:	'This is the menu'

Figure 2: A parallel example from the LDC BOLT Arabizi dataset, written in Latin script (source) and converted to Arabic (target) semi-manually. Some source-side segments (in red) are removed by annotators; we use the version without such segments (filtered) for our task. The annotators also standardize spacing on the target side, which results in difference with the source (in blue).

Arabic We use the LDC BOLT Phase 2 corpus (Bies et al., 2014; Song et al., 2014) for training and testing the Arabic transliteration models (Figure 2). The corpus consists of short SMS and chat in Egyptian Arabic represented using Latin script (*Arabizi*). The corpus is fully parallel: each message is automatically converted into the standardized dialectal Arabic orthography (CODA; Habash et al., 2012) and then manually corrected by human annotators. We split and preprocess the data according to Ryskina et al. (2020), discarding the target (native script) and source (romanized) parallel sentences to create the source and target monolingual training splits respectively.

Russian We use the romanized Russian dataset collected by Ryskina et al. (2020), augmented with the monolingual Cyrillic data from the Taiga corpus of Shavrina and Shapovalova (2017) (Figure 3). The romanized data is split into training, validation, and test portions, and all validation and test sentences are converted to Cyrillic by native speaker annotators. Both the romanized and the native-script sequences are collected from public posts and comments on a Russian social network `vk.com`, and they are on average 3 times longer than the messages in the Arabic dataset (Table 1). However, although both sides were scraped from the same online platform, the relevant Taiga data is collected primarily from political discussion groups, so there is still a substantial domain mismatch between the source and target sides of the data.

[2]Links to download the corpora and other data sources discussed in this section can be found in Appendix A.

	Train (source)		Train (target)		Validation		Test	
	Sent.	Char.	Sent.	Char.	Sent.	Char.	Sent.	Char.
Romanized Arabic	5K	104K	49K	935K	301	8K	1K	20K
Romanized Russian	5K	319K	307K	111M	227	15K	1K	72K
Romanized Kannada	10K	1M	679K	64M	100	11K	100	10K
Serbian→Bosnian	160K	9M	136K	9M	16K	923K	100	9K
Bosnian→Serbian	136K	9M	160K	9M	16K	908K	100	10K

Table 1: Dataset splits for each task and language. The source and target train data are monolingual, and the validation and test sentences are parallel. For the informal romanization task, the source and target sides correspond to the Latin and the original script respectively. For the translation task, the source and target sides correspond to source and target languages. The validation and test character statistics are reported for the source side.

Annotated	
Source:	proishodit s prirodoy 4to to very very bad
Filtered:	proishodit s prirodoy 4to to <...>
Target:	происходит с природой что-то <...>
Gloss:	'Something very very bad is happening to the environment'
Monolingual	
Source:	—
Target:	это видеоролики со съезда партии"Единая Россия"
Gloss:	'These are the videos from the "United Russia" party congress'

Figure 3: **Top:** A parallel example from the romanized Russian dataset. We use the filtered version of the romanized (source) sequences, removing the segments the annotators were unable to convert to Cyrillic, e.g. code-switched phrases (in red). The annotators also standardize minor spelling variation such as hyphenation (in blue). **Bottom:** a monolingual Cyrillic example from the `vk.com` portion of the Taiga corpus, which mostly consists of comments in political discussion groups.

Target:	ಮೂಲ ಸಾಕೆಟ್‌ನಲ್ಲಿ DDR3ಯನ್ನು ಬಳಸಲು
Source:	moola saaketnalli ddr3yannu balasalu
Gloss:	'to use DDR3 in the source circuit'

Figure 4: A parallel example from the Kannada portion of the Dakshina dataset. The Kannada script data (target) is scraped from Wikipedia and manually converted to Latin (source) by human annotators. Foreign target-side characters (in red) get preserved in the annotation but our preprocessing replaces them with UNK on the target side.

Serbian:	свако има право на живот, слободу и безбедност личности.
Bosnian:	svako ima pravo na život, slobodu i osobnu sigurnost.
Gloss:	'Everyone has the right to life, liberty and security of person.'

Figure 5: A parallel example from the Serbian–Cyrillic and Bosnian–Latin UDHR. The sequences are not entirely parallel on character level due to paraphrases and non-cognate translations (in blue).

Kannada Our Kannada data (Figure 4) is taken from the Dakshina dataset (Roark et al., 2020), a large collection of native-script text from Wikipedia for 12 South Asian languages. Unlike the Russian and Arabic data, the romanized portion of Dakshina is not scraped directly from the users' online communication, but instead elicited from native speakers given the native-script sequences. Because of this, all romanized sentences in the data are parallel: we allocate most of them to the source side training data, discarding their original script counterparts, and split the remaining annotated ones between validation and test.

3.2 Related language translation

Following prior work (Pourdamghani and Knight, 2017; Yang et al., 2018; He et al., 2020), we train our unsupervised models on the monolingual data from the Leipzig corpora (Goldhahn et al., 2012). We reuse the non-parallel training and synthetic parallel validation splits of Yang et al. (2018), who generated their parallel data using the Google Translation API. Rather than using their synthetic test set, we opt to test on natural parallel data from the Universal Declaration of Human Rights (UDHR), following Pourdamghani and Knight (2017).

We manually sentence-align the Serbian–

Cyrillic and Bosnian–Latin declaration texts and follow the preprocessing guidelines of Pourdamghani and Knight (2017). Although we strive to approximate the training and evaluation setup of their work for fair comparison, there are some discrepancies: for example, our manual alignment of UDHR yields 100 sentence pairs compared to 104 of Pourdamghani and Knight (2017). We use the data to train the translation models in both directions, simply switching the source and target sides from Serbian to Bosnian and vice versa.

3.3 Inductive bias

As discussed in §1, the WFST models are less powerful than the seq2seq models; however, they are also more structured, which we can use to introduce inductive bias to aid unsupervised training. Following Ryskina et al. (2020), we introduce informative priors on character substitution operations (for a description of the WFST parameterization, see §4.1). The priors reflect the visual and phonetic similarity between characters in different alphabets and are sourced from human-curated resources built with the same concepts of similarity in mind. For all tasks and languages, we collect phonetically similar character pairs from the phonetic keyboard layouts (or, in case of the translation task, from the default Serbian keyboard layout, which is phonetic in nature due to the dual orthography standard of the language). We also add some visually similar character pairs by automatically pairing all symbols that occur in both source and target alphabets (same Unicode codepoints). For Russian, which exhibits a greater degree of visual similarity than Arabic or Kannada, we also make use of the Unicode confusables list (different Unicode codepoints but same or similar glyphs).[3]

It should be noted that these automatically generated informative priors also contain noise: keyboard layouts have spurious mappings because each symbol must be assigned to exactly one key in the QWERTY layout, and Unicode-constrained visual mappings might prevent the model from learning correspondences between punctuation symbols (e.g. Arabic question mark ؟ → ?).

3.4 Preprocessing

We lowercase and segment all sequences into characters as defined by Unicode codepoints, so diacritics and non-printing characters like ZWJ are also treated as separate vocabulary items. To filter out foreign or archaic characters and rare diacritics, we restrict the alphabets to characters that cover 99% of the monolingual training data. After that, we add any standard alphabetical characters and numerals that have been filtered out back into the source and target alphabets. All remaining filtered characters are replaced with a special UNK symbol in all splits except for the target-side test.

[3]Links to the keyboard layouts and the confusables list can be found in Appendix A.

4 Methods

We perform our analysis using the finite-state and seq2seq models from prior work and experiment with two joint decoding strategies, reranking and product of experts. Implementation details and hyperparameters are described in Appendix B.

4.1 Base models

Our finite-state model is the WFST cascade introduced by Ryskina et al. (2020). The model is composed of a character-level n-gram language model and a script conversion transducer (emission model), which supports one-to-one character substitutions, insertions, and deletions. Character operation weights in the emission model are parameterized with multinomial distributions, and similar character mappings (§3.3) are used to create Dirichlet priors on the emission parameters. To avoid marginalizing over sequences of infinite length, a fixed limit is set on the delay of any path (the difference between the cumulative number of insertions and deletions at any timestep). Ryskina et al. (2020) train the WFST using stochastic stepwise EM (Liang and Klein, 2009), marginalizing over all possible target sequences and their alignments with the given source sequence. To speed up training, we modify their training procedure towards 'hard EM': given a source sequence, we predict the most probable target sequence under the model, marginalize over alignments and then update the parameters. Although the unsupervised WFST training is still slow, the stepwise training procedure is designed to converge using fewer data points, so we choose to train the WFST model only on the 1,000 shortest source-side training sequences (500 for Kannada).

Our default seq2seq model is the unsupervised neural machine translation (UNMT) model of Lample et al. (2018, 2019) in the parameterization of He et al. (2020). The model consists of an

	Arabic			Russian			Kannada		
	CER	WER	BLEU	CER	WER	BLEU	CER	WER	BLEU
WFST	.405	.86	2.3	.202	.58	14.8	.359	**.71**	12.5
Seq2Seq	.571	.85	4.0	.229	**.38**	**48.3**	.559	.79	11.3
Reranked WFST	**.398**	.85	2.8	.195	.57	16.1	**.358**	**.71**	12.5
Reranked Seq2Seq	.538	**.82**	**4.6**	.216	.39	45.6	.545	.78	**12.6**
Product of experts	.470	.88	2.5	**.178**	.50	22.9	.543	.93	7.0

Table 2: Character and word error rates (lower is better) and BLEU scores (higher is better) for the romanization decipherment task. **Bold** indicates best per column. Model combinations mostly interpolate between the base models' scores, although reranking yields minor improvements in character-level and word-level metrics for the WFST and seq2seq respectively. **Note:** base model results are not intended as a direct comparison between the WFST and seq2seq, since they are trained on different amounts of data.

	srp→bos			bos→srp		
	CER	WER	BLEU	CER	WER	BLEU
WFST	**.314**	.50	25.3	.319	.52	25.5
Seq2Seq	.375	.49	34.5	.395	.49	36.3
Reranked WFST	**.314**	.49	26.3	**.317**	.50	28.1
Reranked Seq2Seq	.376	**.48**	35.1	.401	**.47**	37.0
Product of experts	.329	.54	24.4	.352	.66	20.6
(Pourdamghani and Knight, 2017)	—	—	**42.3**	—	—	**39.2**
(He et al., 2020)	.657	.81	5.6	.693	.83	4.7

Table 3: Character and word error rates (lower is better) and BLEU scores (higher is better) for the related language translation task. **Bold** indicates best per column. The WFST and the seq2seq have comparable CER and WER despite the WFST being trained on up to 160x less source-side data (§4.1). While none of our models achieve the scores reported by Pourdamghani and Knight (2017), they all substantially outperform the subword-level model of He et al. (2020). **Note:** base model results are not intended as a direct comparison between the WFST and seq2seq, since they are trained on different amounts of data.

LSTM (Hochreiter and Schmidhuber, 1997) encoder and decoder with attention, trained to map sentences from each domain into a shared latent space. Using a combined objective, the UNMT model is trained to denoise, translate in both directions, and discriminate between the latent representation of sequences from different domains. Since the sufficient amount of balanced data is crucial for the UNMT performance, we train the seq2seq model on all available data on both source and target sides. Additionally, the seq2seq model decides on early stopping by evaluating on a small parallel validation set, which our WFST model does not have access to.

The WFST model treats the target and source training data differently, using the former to train the language model and the latter for learning the emission parameters, while the UNMT model is trained to translate in both directions simultaneously. Therefore, we reuse the same seq2seq model for both directions of the translation task, but train a separate finite-state model for each direction.

4.2 Model combinations

The simplest way to combine two independently trained models is reranking: using one model to produce a list of candidates and rescoring them according to another model. To generate candidates with a WFST, we apply the n–shortest paths algorithm (Mohri and Riley, 2002). It should be noted that the n–best list might contain duplicates since each path represents a specific source–target character alignment. The length constraints encoded in the WFST also restrict its capacity as a reranker: beam search in the UNMT model may produce hypotheses too short or long to have a non-zero

Input	свако има право да слободно учествује у културном животу заједнице, да ужива у уметности и да учествује у научном напретку и у добробити која отуда проистиче.
Ground truth	svako ima pravo da slobodno sudjeluje u kulturnom životu zajednice, da uživa u umjetnosti i da učestvuje u znanstvenom napretku i u njegovim koristima.
WFST	svako ima pravo da slobodno učestvuje u kulturnom životu sjednice , da uživa u metnosti i da učestvuje u naučnom napretku i u dobrobiti koja otuda prističe .
Reranked WFST	svako ima pravo da slobodno učestvuje u kulturnom životu sjednice , da uživa u metnosti i da učestvuje u naučnom napretku i u dobrobiti koja otuda prističe .
Seq2Seq	svako ima pravo da slobodno učestvuje u kulturnom životu zajednice , da učestvuje u naučnom napretku i u dobrobiti koja otuda proističe .
Reranked Seq2Seq	svako ima pravo da slobodno učestvuje u kulturnom životu zajednice , da uživa u umjetnosti i da učestvuje u naučnom napretku i u dobrobiti koja otuda proističe
Product of experts	svako ima pravo da slobodno učestvuje u kulturnom za u sajednice , da živa u umjetnosti i da učestvuje u naučnom napretku i u dobroj i koja otuda proisti
Subword Seq2Seq	sami ima pravo da slobodno utiče na srpskom nivou vlasti da razgovaraju u bosne i da djeluje u međunarodnom turizmu i na buducnosti koja muža decisno .

Table 4: Different model outputs for a srp→bos translation example. Prediction errors are highlighted in red. Correctly transliterated segments that do not match the ground truth (e.g. due to paraphrasing) are shown in yellow. Here the WFST errors are substitutions or deletions of individual characters, while the seq2seq drops entire words from the input (§5 #4). The latter problem is solved by reranking with a WFST for this example. The seq2seq model with subword tokenization (He et al., 2020) produces mostly hallucinated output (§5 #2). Example outputs for all other datasets can be found in the Appendix.

probability under the WFST.

Our second approach is a product-of-experts-style joint decoding strategy (Hinton, 2002): we perform beam search on the WFST lattice, reweighting the arcs with the output distribution of the seq2seq decoder at the corresponding timestep. For each partial hypothesis, we keep track of the WFST state s and the partial input and output sequences $x_{1:k}$ and $y_{1:t}$.[4] When traversing an arc with input label $i \in \{x_{k+1}, \epsilon\}$ and output label o, we multiply the arc weight by the probability of the neural model outputting o as the next character: $p_{\text{seq2seq}}(y_{t+1} = o|x, y_{1:t})$. Transitions with $o = \epsilon$ (i.e. deletions) are not rescored by the seq2seq. We group hypotheses by their consumed input length k and select n best extensions at each timestep.

4.3 Additional baselines

For the translation task, we also compare to prior unsupervised approaches of different granularity: the deep generative style transfer model of He et al. (2020) and the character- and word-level WFST decipherment model of Pourdamghani and Knight (2017). The former is trained on the same training set tokenized into subword units (Sennrich et al., 2016), and we evaluate it on our UDHR test set for fair comparison. While the train and test data of Pourdamghani and Knight (2017) also use the same respective sources, we cannot account for tokenization differences that could affect the scores reported by the authors.

[4]Due to insertions and deletions in the emission model, k and t might differ; epsilon symbols are not counted.

5 Results and analysis

Tables 2 and 3 present our evaluation of the two base models and three decoding-time model combinations on the romanization decipherment and related language translation tasks respectively. For each experiment, we report character error rate, word error rate, and BLEU (see Appendix C). The results for the base models support what we show later in this section: the seq2seq model is more likely to recover words correctly (higher BLEU, lower WER), while the WFST is more faithful on character level and avoids word-level substitution errors (lower CER). Example predictions can be found in Table 4 and in the Appendix.

Our further qualitative and quantitative findings are summarized in the following high-level takeaways:

#1: Model combinations still suffer from search issues. We would expect the combined decoding to discourage all errors common under one model but not the other, improving the performance by leveraging the strengths of both model classes. However, as Tables 2 and 3 show, they instead

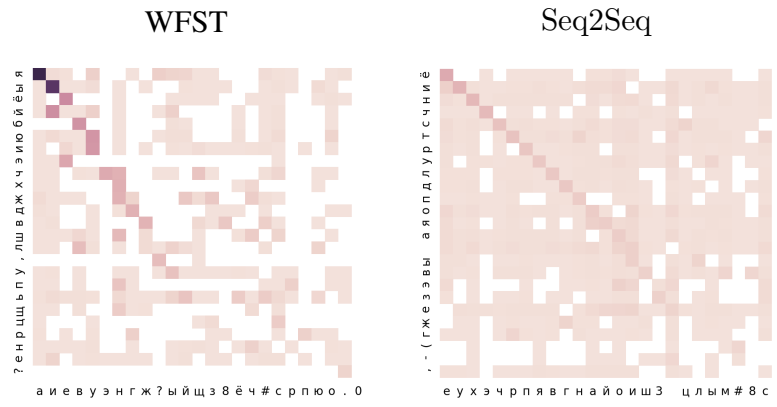

Figure 6: Highest-density submatrices of the two base models' character confusion matrices, computed in the Russian romanization task. White cells represent zero elements. The WFST confusion matrix (left) is noticeably sparser than the seq2seq one (right), indicating more repetitive errors. # symbol stands for UNK.

mostly interpolate between the scores of the two base models. In the reranking experiments, we find that this is often due to the same base model error (e.g. the seq2seq model hallucinating a word mid-sentence) repeating across all the hypotheses in the final beam. This suggests that successful reranking would require a much larger beam size or a diversity-promoting search mechanism.

Interestingly, we observe that although adding a reranker on top of a decoder does improve performance slightly, the gain is only in terms of the metrics that the base decoder is already strong at—character-level for reranked WFST and word-level for reranked seq2seq—at the expense of the other scores. Overall, none of our decoding strategies achieves best results across the board, and no model combination substantially outperforms both base models in any metric.

#2: Character tokenization boosts performance of the neural model. In the past, UNMT-style models have been applied to various unsupervised sequence transduction problems. However, since these models were designed to operate on word or subword level, prior work assumes the same tokenization is necessary. We show that for the tasks allowing character-level framing, such models in fact respond extremely well to character input.

Table 3 compares the UNMT model trained on characters with the seq2seq style transfer model of He et al. (2020) trained on subword units. The original paper shows improvement over the UNMT baseline in the same setting, but simply switching to character-level tokenization without any other changes results in a 30 BLEU points gain for either direction. This suggests that the tokenization choice could act as an inductive bias for seq2seq models, and character-level framing could be useful even for tasks that are not truly character-level.

This observation also aligns with the findings of the recent work on language modeling complexity (Park et al., 2021; Mielke et al., 2019). For many languages, including several Slavic ones related to the Serbian–Bosnian pair, a character-level language model yields lower surprisal than the one trained on BPE units, suggesting that the effect might also be explained by the character tokenization making the language easier to language-model.

#3: WFST model makes more repetitive errors. Although two of our evaluation metrics, CER and WER, are based on edit distance, they do not distinguish between the different types of edits (substitutions, insertions and deletions). Breaking them down by the edit operation, we find that while both models favor substitutions on both word and character levels, insertions and deletions are more frequent under the neural model (43% vs. 30% of all edits on the Russian romanization task). We also find that the character substitution choices of the neural model are more context-dependent: while the total counts of substitution errors for the two models are comparable, the WFST is more likely to repeat the same few substitutions per character type. This is illustrated by Figure 6, which visualizes the most populated submatrices of the confusion matrices for the same task as heatmaps. The WFST confusion matrix is noticeably more sparse, with the same few substitutions occurring much more frequently than others: for example, WFST often mistakes я for a and rarely for other characters, while the neural model's substitutions of я are distributed closer to uniform. This suggests that the WFST errors might be easier to correct with rule-based postprocessing. Interestingly, we did not observe the same effect for the translation task, likely due to a more constrained nature of the orthography conversion.

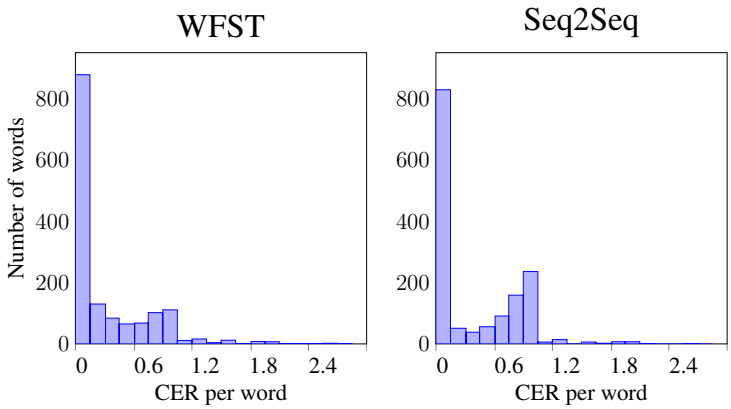

Figure 7: Character error rate per word for the WFST (left) and seq2seq (right) bos→srp translation outputs. The predictions are segmented using Moses tokenizer (Koehn et al., 2007) and aligned to ground truth with word-level edit distance. The increased frequency of CER=1 for the seq2seq model as compared to the WFST indicates that it replaces entire words more often.

#4: Neural model is more sensitive to data distribution shifts. The language model aiming to replicate its training data distribution could cause the output to deviate from the input significantly. This could be an artifact of a domain shift, such as in Russian, where the LM training data came from a political discussion forum: the seq2seq model frequently predicts unrelated domain-specific proper names in place of very common Russian words, e.g. жизнь [žizn, 'life'] → Зюганов [Zjuganov, 'Zyuganov (politician's last name)'] or это [èto, 'this'] → Единая Россия [Edinaja Rossija, 'United Russia (political party)'], presumably distracted by the shared first character in the romanized version. To quantify the effect of a mismatch between the train and test data distributions in this case, we inspect the most common word-level substitutions under each decoding strategy, looking at all substitution errors covered by the 1,000 most frequent substitution 'types' (ground truth–prediction word pairs) under the respective decoder. We find that 25% of the seq2seq substitution errors fall into this category, as compared to merely 3% for the WFST—notable given the relative proportion of in-vocabulary words in the models' outputs (89% for UNMT vs. 65% for WFST).

Comparing the error rate distribution across output words for the translation task also supports this observation. As can be seen from Figure 7, the seq2seq model is likely to either predict the word correctly (CER of 0) or entirely wrong (CER of 1), while the the WFST more often predicts the word partially correctly—examples in Table 4 illustrate this as well. We also see this in the Kannada outputs: WFST typically gets all the consonants right but makes mistakes in the vowels, while the seq2seq tends to replace the entire word.

6 Conclusion

We perform comparative error analysis in finite-state and seq2seq models and their combinations for two unsupervised character-level tasks, informal romanization decipherment and related language translation. We find that the two model types tend towards different errors: seq2seq models are more prone to word-level errors caused by distributional shifts while WFSTs produce more character-level noise despite the hard alignment constraints.

Despite none of our simple decoding-time combinations substantially outperforming the base models, we believe that combining neural and finite-state models to harness their complementary advantages is a promising research direction. Such combinations might involve biasing seq2seq models towards WFST-like behavior via pretraining or directly encoding constraints such as hard alignment or monotonicity into their parameterization (Wu et al., 2018; Wu and Cotterell, 2019). Although recent work has shown that the Transformer can learn to perform character-level transduction without such biases in a supervised setting (Wu et al., 2021), exploiting the structured nature of the task could be crucial for making up for the lack of large parallel corpora in low-data and/or unsupervised scenarios. We hope that our analysis provides insight into leveraging the strengths of the two approaches for modeling character-level phenomena in the absence of parallel data.

Acknowledgments

The authors thank Badr Abdullah, Deepak Gopinath, Junxian He, Shruti Rijhwani, and Stas Kashepava for helpful discussion, and the anonymous reviewers for their valuable feedback.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

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 the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. http://www.openfst.org.

Sowmya V. B., Monojit Choudhury, Kalika Bali, Tirthankar Dasgupta, and Anupam Basu. 2010. Resource creation for training and testing of transliteration systems for Indian languages. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Ann Bies, Zhiyi Song, Mohamed Maamouri, Stephen Grimes, Haejoong Lee, Jonathan Wright, Stephanie Strassel, Nizar Habash, Ramy Eskander, and Owen Rambow. 2014. Transliteration of Arabizi into Arabic orthography: Developing a parallel annotated Arabizi-Arabic script SMS/chat corpus. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 93–103, Doha, Qatar. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Kareem Darwish. 2014. Arabizi detection and conversion to Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 217–224, Doha, Qatar. Association for Computational Linguistics.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey. European Language Resources Association (ELRA).

Kyle Gorman. 2016. Pynini: A Python library for weighted finite-state grammar compilation. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 75–80, Berlin, Germany. Association for Computational Linguistics.

Nizar Habash, Mona Diab, and Owen Rambow. 2012. Conventional orthography for dialectal Arabic. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 711–718, Istanbul, Turkey. European Language Resources Association (ELRA).

Bradley Hauer, Ryan Hayward, and Grzegorz Kondrak. 2014. Solving substitution ciphers with combined language models. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2314–2325, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020. A probabilistic formulation of unsupervised text style transfer. In *International Conference on Learning Representations*.

Felix Hieber and Stefan Riezler. 2015. Bag-of-words forced decoding for cross-lingual information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1182, Denver, Colorado. Association for Computational Linguistics.

G. E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Cibu Johny, Lawrence Wolf-Sonkin, Alexander Gutkin, and Brian Roark. 2021. Finite-state script normalization and processing utilities: The Nisaba Brahmic library. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 14–23, Online. Association for Computational Linguistics.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 499–506, Sydney, Australia. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *International Conference on Learning Representations*.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 611–619, Boulder, Colorado. Association for Computational Linguistics.

Chu-Cheng Lin, Hao Zhu, Matthew R. Gormley, and Jason Eisner. 2019. Neural finite-state transducers: Beyond rational relations. 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 272–283, Minneapolis, Minnesota. Association for Computational Linguistics.

Nikola Ljubešić and Filip Klubička. 2014. {bs,hr,sr}WaC - web corpora of Bosnian, Croatian and Serbian. In *Proceedings of the 9th Web as Corpus Workshop (WaC-9)*, pages 29–35, Gothenburg, Sweden. Association for Computational Linguistics.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 49–57, Vancouver. Association for Computational Linguistics.

Sabrina J. Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. 2019. What kind of language is hard to language-model? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4975–4989, Florence, Italy. Association for Computational Linguistics.

Mehryar Mohri. 2009. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer.

Mehryar Mohri and Michael Riley. 2002. An efficient algorithm for the n-best-strings problem. In *Seventh International Conference on Spoken Language Processing*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Hyunji Hayley Park, Katherine J. Zhang, Coleman Haley, Kenneth Steimel, Han Liu, and Lane Schwartz. 2021. Morphology matters: A multilingual language modeling analysis. *Transactions of the Association for Computational Linguistics*, 9:261–276.

Martin Paulsen. 2014. Translit: Computer-mediated digraphia on the Runet. *Digital Russia: The Language, Culture and Politics of New Media Communication*.

Nima Pourdamghani and Kevin Knight. 2017. Deciphering related languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2513–2518, Copenhagen, Denmark. Association for Computational Linguistics.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2009. Learning phoneme mappings for transliteration without parallel data. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 37–45, Boulder, Colorado. Association for Computational Linguistics.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66, Jeju Island, Korea. Association for Computational Linguistics.

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 12th Language Resources and Evaluation Conference*, pages 2413–2423, Marseille, France. European Language Resources Association.

Maria Ryskina, Matthew R. Gormley, and Taylor Berg-Kirkpatrick. 2020. Phonetic and visual priors for

decipherment of informal Romanization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8308–8319, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Tatiana Shavrina and Olga Shapovalova. 2017. To the methodology of corpus construction for machine learning: Taiga syntax tree corpus and parser. In *Proc. CORPORA 2017 International Conference*, pages 78–84, St. Petersburg.

Zhiyi Song, Stephanie Strassel, Haejoong Lee, Kevin Walker, Jonathan Wright, Jennifer Garland, Dana Fore, Brian Gainor, Preston Cabe, Thomas Thomas, Brendan Callahan, and Ann Sawyer. 2014. Collecting natural SMS and chat conversations in multiple languages: The BOLT phase 2 corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1699–1704, Reykjavik, Iceland. European Language Resources Association (ELRA).

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1530–1537, Florence, Italy. Association for Computational Linguistics.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2021. Applying the transformer to character-level transduction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1901–1907, Online. Association for Computational Linguistics.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438, Brussels, Belgium. Association for Computational Linguistics.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P. Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *NeurIPS*, pages 7298–7309.

A Data download links

The romanized Russian and Arabic data and preprocessing scripts can be downloaded here. This repository also contains the relevant portion of the Taiga dataset, which can be downloaded in full at this link. The romanized Kannada data was downloaded from the Dakshina dataset.

The scripts to download the Serbian and Bosnian Leipzig corpora data can be found here. The UDHR texts were collected from the corresponding pages: Serbian, Bosnian.

The keyboard layouts used to construct the phonetic priors are collected from the following sources: Arabic 1, Arabic 2, Russian, Kannada, Serbian. The Unicode confusables list used for the Russian visual prior can be found here.

B Implementation

WFST We reuse the unsupervised WFST implementation of Ryskina et al. (2020),[5] which utilizes the OpenFst (Allauzen et al., 2007) and OpenGrm (Roark et al., 2012) libraries. We use the default hyperparameter settings described by the authors (see Appendix B in the original paper). We keep the hyperparameters unchanged for the translation experiment and set the maximum delay value to 2 for both translation directions.

UNMT We use the PyTorch UNMT implementation of He et al. (2020)[6] which incorporates improvements introduced by Lample et al. (2019) such as the addition of a max-pooling layer. We use a single-layer LSTM (Hochreiter and Schmidhuber, 1997) with hidden state size 512 for both the encoder and the decoder and embedding dimension 128. For the denoising autoencoding loss, we adopt the default noise model and hyperparameters as described by Lample et al. (2018). The autoencoding loss is annealed over the first 3 epochs. We predict the output using greedy decoding and set the maximum output length equal to the length of the input sequence. Patience for early stopping is set to 10.

Model combinations Our joint decoding implementations rely on PyTorch and the Pynini finite-state library (Gorman, 2016). In reranking, we rescore $n = 5$ best hypotheses produced using beam search and n–shortest path algorithm for the UNMT and WFST respectively. Product of experts decoding is also performed with beam size 5.

C Metrics

The character error rate (CER) and word error rate (WER) as measured as the Levenshtein distance between the hypothesis and reference divided by reference length:

$$\mathrm{ER}(h, r) = \frac{dist(h, r)}{len(r)}$$

with both the numerator and the denominator measured in characters and words respectively.

We report BLEU-4 score (Papineni et al., 2002), measured using the Moses toolkit script.[7] For both BLEU and WER, we split sentences into words using the Moses tokenizer (Koehn et al., 2007).

[5] https://github.com/ryskina/romanization-decipherment

[6] https://github.com/cindyxinyiwang/deep-latent-sequence-model

[7] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

Input	kongress ne odobril biudjet dlya osuchestvleniye "bor'bi s kommunizmom" v yuzhniy amerike.	
Ground truth	конгресс не одобрил бюджет для осуществления "борьбы с коммунизмом" в южной америке.	kongress ne odobril bjudžet dlja osuščestvlenija "bor'by s kommunizmom" v južnoj amerike.
WFST	конгресс не одобрил виудет дла осусчествлениые "бор#би с коммунизмом" в уузнани америке.	kongress ne odobril viudet dla osuščestvleniye "bor#bi s kommunizmom" v uuznani amerike.
Reranked WFST	конгресс не одобрил видет дела осусчествлениые "бор#би с коммунизмом" в уузнани америке.	kongress ne odobril videt dela osuščestvleniye "bor#bi s kommunizmom" v uuznani amerike.
Seq2Seq	конгресс не одобрил бы удивительно с коммунизмом" в южный америке.	kongress ne odobril by udivitel'no s kommunizmom" v južnyj amerike.
Reranked Seq2Seq	конгресс не одобрил бюджет для осуществление "борьбы с коммунизмом" в южный америке.	kongress ne odobril bjudžet dlja osuščestvlenie "bor'by s kommunizmom" v južnyj amerike.
Product of experts	конгресс не одобрил бидет для а осуществлениые "борьбы с коммунизмом" в уузник амери	kongress ne odobril bidet dlja a osuščestvleniye "bor'by s kommunizmom" v uuznik ameri

Table 5: Different model outputs for a Russian transliteration example (left column—Cyrillic, right—scientific transliteration). Prediction errors are shown in red. Correctly transliterated segments that do not match the ground truth because of spelling standardization in annotation are in yellow. # stands for UNK.

Input	ana h3dyy 3lek bokra 3la 8 kda	
Ground truth	انا حأعدي عليك بكرة على 8كده	AnA H>Edy Elyk bkrp ElY 8 kdh
WFST	انا حد يي لك بكر لأ 8كده	AnA Hd yy lk bkr l> 8 kdh
Reranked WFST	انا حد يي لك بكر لأ 8كده	AnA Hd yy lk bkr l> 8 kdh
Seq2Seq	انا بأدي أخلك حر أول 1كده	AnA b>dy >xlk Hr >wl 1 kdh
Reranked Seq2Seq	انا بأدي أخلك حر أول 1كده	AnA b>dy >xlk Hr >wl 1 kdh
Product of experts	انا دي لك ب كرا ألا 8كده	AnA dy lk b krA > lA 8 kdh

Table 6: Different model outputs for an Arabizi transliteration example (left column—Arabic, right—Buckwalter transliteration). Prediction errors are highlighted in red in the romanized versions. Correctly transliterated segments that do not match the ground truth because of spelling standardization during annotation are highlighted in yellow.

Input	kshullaka baalina avala horaatavannu adu vivarisuttade.	
Ground truth	ಕ್ಷುಲ್ಲಕ ಬಾಳಿನ ಅವಳ ಹೋರಾಟವನ್ನು ಅದು ವಿವರಿಸುತ್ತದೆ.	kṣullaka bāḷina avaḷa hōrāṭavannu adu vivarisuttade.
WFST	ಕುಹೂಲ್ಲಾಖೆ ಬಾಲಿನು ವಾಳ ಹೊರತಾವನ್ನು ಆದು ವಿವರಿಸುತ್ತದೆ.	kuhūllākhe bālinu vāḷa horatāvannu ādu vivarisuttade.
Reranked WFST	ಕುಹೂಲ್ಲಾಖೆ ಬಾಲಿನ ವಾಳು ಹೊರತಾವನ್ನು ಆದು ವಿವರಿಸುತ್ತದೆ.	kuhūllākhe bālina vāḷu horatāvannu ādu vivarisuttade.
Seq2Seq	ಕಳುಹುಳ್ಳ ಬಾವಿಂಗ್ ಇಲ್ಲವೇ ಹೋರಾಟವನ್ನು ಇದು ವಿವರಿಸುತ್ತದೆ.	kaḷuhuḷḷa bāviṁg illavē hōrāṭavannu idu vivarisuttade.
Reranked Seq2Seq	ಕಳುಹುಳ್ಳ ಬಾವಿಂತ ಇಲ್ಲವೇ ಹೋರಾಟವನ್ನು ಇದು ವಿವರಿಸುತ್ತದೆ.	kaḷuhuḷḷa bāviṁta illavē hōrāṭavannu idu vivarisuttade.
Product of experts	ಕಳ್ಳ ಬಾಕಲಿನ್ನ ವಾಲಾ ಹೋರಾಟತ್ವಾನ್ನು ದು ವಿವಾರಿಸುತ್ತದ	kaḷḷa bākalinna vālā hōrāṭatvānnu du vivārisuttada

Table 7: Different model outputs for a Kannada transliteration example (left column—Kannada, right—ISO 15919 transliterations). The ISO romanization is generated using the Nisaba library (Johny et al., 2021). Prediction errors are highlighted in red in the romanized versions.