Phonetic and Visual Priors for Decipherment of Informal Romanization

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

Abstract

Informal romanization is an idiosyncratic process used by humans in informal digital communication to encode non-Latin script languages into Latin character sets found on common keyboards. Character substitution choices differ between users but have been shown to be governed by the same main principles observed across a variety of languages—namely, character pairs are often associated through phonetic or visual similarity. We propose a noisy-channel WFST cascade model for deciphering the original non-Latin script from observed romanized text in an unsupervised fashion. We train our model directly on romanized data from two languages: Egyptian Arabic and Russian. We demonstrate that adding inductive bias through phonetic and visual priors on character mappings substantially improves the model's performance on both languages, yielding results much closer to the supervised skyline. Finally, we introduce a new dataset of romanized Russian, collected from a Russian social network website and partially annotated for our experiments.[1]

1 Introduction

Written online communication poses a number of challenges for natural language processing systems, including the presence of neologisms, code-switching, and the use of non-standard orthography. One notable example of orthographic variation in social media is *informal romanization*[2]—speakers of languages written in non-Latin alphabets encoding their messages in Latin characters, for convenience or due to technical constraints (improper rendering of native script or keyboard layout incompatibility). An example of such a sentence can be found in Figure 2. Unlike named entity transliteration where the change of script represents the change of language, here Latin characters serve as an intermediate symbolic representation to be decoded by another speaker of the same source language, calling for a completely different transliteration mechanism: instead of expressing the pronunciation of the word according to the phonetic rules of another language, informal transliteration can be viewed as a substitution cipher, where each source character is replaced with a similar Latin character.

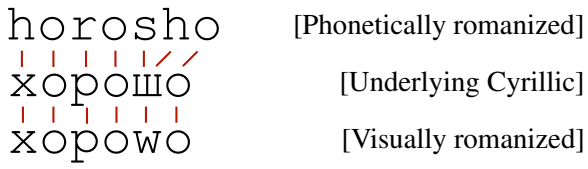

Figure 1: Example transliterations of a Russian word хорошо [horošo, 'good'] (middle) based on phonetic (top) and visual (bottom) similarity, with character alignments displayed. The phonetic-visual dichotomy gives rise to one-to-many mappings such as *ш* /ʃ/ → *sh* / *w*.

In this paper, we focus on decoding informally romanized texts back into their original scripts. We view the task as a decipherment problem and propose an unsupervised approach, which allows us to save annotation effort since parallel data for informal transliteration does not occur naturally. We propose a weighted finite-state transducer (WFST) cascade model that learns to decode informal romanization without parallel text, relying only on transliterated data and a language model over the original orthography. We test it on two languages, Egyptian Arabic and Russian, collecting our own dataset of romanized Russian from a Russian social network website vk.com.

[1]The code and data are available at https://github.com/ryskina/romanization-decipherment

[2]Our focus on *informal* transliteration excludes formal settings such as pinyin for Mandarin where transliteration conventions are well established.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8308–8319
July 5 - 10, 2020.

4to mowet bit' ly4we?	[Romanized]
Что может быть лучше?	[Latent Cyrillic]
Čto možet byt' lučše?	[Scientific]
/ʃto ˈmoʒɨt bɨtʲ ˈlutʃʃɨ/	[IPA]
What can be better?	[Translated]

Figure 2: Example of an informally romanized sentence from the dataset presented in this paper, containing a many-to-one mapping *ж* / *ш* → *w*. Scientific transliteration, broad phonetic transcription, and translation are not included in the dataset and are presented for illustration only.

Since informal transliteration is not standardized, converting romanized text back to its original orthography requires reasoning about the specific user's transliteration preferences and handling many-to-one (Figure 2) and one-to-many (Figure 1) character mappings, which is beyond traditional rule-based converters. Although user behaviors vary, there are two dominant patterns in informal romanization that have been observed independently across different languages, such as Russian (Paulsen, 2014), dialectal Arabic (Darwish, 2014) or Greek (Chalamandaris et al., 2006):

Phonetic similarity: Users represent source characters with Latin characters or digraphs associated with similar phonemes (e.g. *м* /m/ → *m*, *л* /l/ → *l* in Figure 2). This substitution method requires implicitly tying the Latin characters to a phonetic system of an intermediate language (typically, English).

Visual similarity: Users replace source characters with similar-looking symbols (e.g. *ч* /t͡ʃʲ/ → *4*, *у* /u/ → *y* in Figure 2). Visual similarity choices often involve numerals, especially when the corresponding source language phoneme has no English equivalent (e.g. Arabic ع /ʕ/ → 3).

Taking that consistency across languages into account, we show that incorporating these style patterns into our model as priors on the emission parameters—also constructed from naturally occurring resources—improves the decoding accuracy on both languages. We compare the proposed unsupervised WFST model with a supervised WFST, an unsupervised neural architecture, and commercial systems for decoding romanized Russian (*translit*) and Arabic (*Arabizi*). Our unsupervised WFST outperforms the unsupervised neural baseline on both languages.

2 Related work

Prior work on informal transliteration uses supervised approaches with character substitution rules either manually defined or learned from automatically extracted character alignments (Darwish, 2014; Chalamandaris et al., 2004). Typically, such approaches are pipelined: they produce candidate transliterations and rerank them using modules encoding knowledge of the source language, such as morphological analyzers or word-level language models (Al-Badrashiny et al., 2014; Eskander et al., 2014). Supervised finite-state approaches have also been explored (Wolf-Sonkin et al., 2019; Hellsten et al., 2017); these WFST cascade models are similar to the one we propose, but they encode a different set of assumptions about the transliteration process due to being designed for abugida scripts (using consonant-vowel syllables as units) rather than alphabets. To our knowledge, there is no prior unsupervised work on this problem.

Named entity transliteration, a task closely related to ours, is better explored, but there is little unsupervised work on this task as well. In particular, Ravi and Knight (2009) propose a fully unsupervised version of the WFST approach introduced by Knight and Graehl (1998), reframing the task as a decipherment problem and learning cross-lingual phoneme mappings from monolingual data. We take a similar path, although it should be noted that named entity transliteration methods cannot be straightforwardly adapted to our task due to the different nature of the transliteration choices. The goal of the standard transliteration task is to communicate the pronunciation of a sequence in the source language (SL) to a speaker of the target language (TL) by rendering it appropriately in the TL alphabet; in contrast, informal romanization emerges in communication between SL speakers only, and TL is not specified. If we picked any specific Latin-script language to represent TL (e.g. English, which is often used to ground phonetic substitutions), many of the informally romanized sequences would still not conform to its pronunciation rules: the transliteration process is character-level rather than phoneme-level and does not take possible TL digraphs into account (e.g. Russian *сх* /sx/ → *sh*), and it often involves eclectic visual substitution choices such as numerals or punctua-

tion (e.g. Arabic تحت [tHt, 'under'][3] → *ta7t*, Russian для [dlja, 'for'] → *dl9|*).

Finally, another relevant task is translating between closely related languages, possibly written in different scripts. An approach similar to ours is proposed by Pourdamghani and Knight (2017). They also take an unsupervised decipherment approach: the cipher model, parameterized as a WFST, is trained to encode the source language character sequences into the target language alphabet as part of a character-level noisy-channel model, and at decoding time it is composed with a word-level language model of the source language. Recently, the unsupervised neural architectures (Lample et al., 2018, 2019) have also been used for related language translation and similar decipherment tasks (He et al., 2020), and we extend one of these neural models to our character-level setup to serve as a baseline (§5).

3 Methods

We train a character-based noisy-channel model that transforms a character sequence o in the native alphabet of the language into a sequence of Latin characters l, and use it to decode the romanized sequence l back into the original orthography. Our proposed model is composed of separate transition and emission components as discussed in §3.1, similarly to an HMM. However, an HMM assumes a one-to-one alignment between the characters of the observed and the latent sequences, which is not true for our task. One original script character can be aligned to two consecutive Latin characters or vice versa: for example, when a phoneme is represented with a single symbol on one side but with a digraph on the other (Figure 1), or when a character is omitted on one side but explicitly written on the other (e.g. short vowels not written in unvocalized Arabic but written in transliteration, or the Russian soft sign ь representing palatalization being often omitted in the romanized version). To handle those alignments, we introduce insertions and deletions into the emission model and modify the emission transducer to limit the number of consecutive insertions and deletions. In our experiments, we compare the performance of the model with and without informative phonetic and visual similarity priors described in §3.2.

[3]The square brackets following a foreign word show its linguistic transliteration (using the scientific and the Buckwalter schemas for Russian and Arabic respectively) and its English translation.

3.1 Model

If we view the process of romanization as encoding a source sequence o into Latin characters, we can consider each observation l to have originated via o being generated from a distribution $p(o)$ and then transformed to Latin script according to another distribution $p(l|o)$. We can write the probability of the observed Latin sequence as:

$$p(l) = \sum_o p(o;\gamma) \cdot p(l|o;\theta) \cdot p_{\text{prior}}(\theta;\alpha) \quad (1)$$

The first two terms in (1) correspond to the probabilities under the transition model (the language model trained on the original orthography) and the emission model respectively. The third term represents the prior distribution on the emission model parameters through which we introduce human knowledge into the model. Our goal is to learn the parameters θ of the emission distribution with the transition parameters γ being fixed.

We parameterize the emission and transition distributions as weighted finite-state transducers (WFSTs):

Transition WFSA The n-gram weighted finite-state acceptor (WFSA) T represents a character-level n-gram language model of the language in the native script, producing the native alphabet character sequence o with the probability $p(o;\gamma)$. We use the parameterization of Allauzen et al. (2003), with the states encoding conditioning history, arcs weighted by n-gram probabilities, and failure transitions representing backoffs. The role of T is to inform the model of what well-formed text in the original orthography looks like; its parameters γ are learned from a separate corpus and kept fixed during the rest of the training.

Emission WFST The emission WFST S transduces the original script sequence o to a Latin sequence l with the probability $p(l|o;\theta)$. Since there can be multiple paths through S that correspond to the input-output pair (o, l), this probability is summed over all such paths (i.e. is a marginal over all possible monotonic character alignments):

$$p(l|o;\theta) = \sum_e p(l, e|o;\theta) \quad (2)$$

We view each path e as a sequence of edit operations: substitutions of original characters with Latin ones ($c_o \rightarrow c_l$), insertions of Latin characters ($\epsilon \rightarrow c_l$), and deletions of original characters ($c_o \rightarrow \epsilon$). Each arc in S corresponds to one

of the possible edit operations; an arc representing the edit $c_o \rightarrow c_l$ is characterized by the input label c_o, the output label c_l, and the weight $-\log p(c_l|c_o;\theta)$. The emission parameters θ are the multinomial conditional probabilities of the edit operations $p(c_l|c_o)$; we learn θ using the algorithm described in §3.3.

3.2 Phonetic and visual priors

To inform the model of which pairs of symbols are close in the phonetic or visual space, we introduce the priors on the emission parameters, increasing the probability of an original alphabet character being substituted by a similar Latin one. Rather than attempting to operationalize the notions of phonetic or visual similarity, we choose to read the likely mappings between symbols off human-compiled resources that use the same underlying principle: phonetic keyboard layouts and visually confusable symbol lists. Examples of mappings that we encode as priors can be found in Table 1.

Phonetic similarity Since we think of the informal romanization as a cipher, we aim to capture the phonetic similarity between characters based on association rather than on the actual grapheme-to-phoneme mappings in specific words. We approximate it using *phonetic keyboard layouts*, one-to-one mappings built to bring together "similar-sounding" characters in different alphabets. We take the character pairs from a union of multiple layouts for each language, two for Arabic[4] and four for Russian.[5] The main drawback of using keyboard layouts is that they require every character to have a Latin counterpart, so some mappings will inevitably be arbitrary; we compensate for this effect by averaging over several layouts.

Visual similarity The strongest example of visual character similarity would be *homoglyphs*—symbols from different alphabets represented by the same glyph, such as Cyrillic *а* and Latin *a*. The fact that homoglyph pairs can be made indistinguishable in certain fonts has been exploited in phishing attacks, e.g. when Latin characters are replaced by virtually identical Cyrillic ones (Gabrilovich and Gontmakher, 2002). This led the Unicode Consortium to publish a list of symbols and symbol combinations similar enough to be potentially confusing to the human eye (referred to as *confusables*).[6] This list contains not only exact homoglyphs but also strongly homoglyphic pairs such as Cyrillic *IO* and Latin *lO*.

Original	Latin	
	Phon.	Vis.
р /r/	*r*	*p*
б /b/	*b*	*b, 6*
в /v/	*v, w*	*b*
و /w, uː, oː/	*w, u*	—
خ /x/	*k, x*	—

Table 1: Example Cyrillic–Latin and Arabic–Latin mappings encoded in the visual and phonetic priors respectively.

We construct a visual prior for the Russian model from all Cyrillic–Latin symbol pairs in the Unicode confusables list.[7] Although this list does not cover more complex visual associations used in informal romanization, such as partial similarity (Arabic Alif with Hamza أ → 2 due to Hamza ء resembling an inverted 2) or similarity conditioned on a transformation such as reflection (Russian *л* → *v*), it makes a sensible starting point. However, this restrictive definition of visual similarity does not allow us to create a visual prior for Arabic—the two scripts are dissimilar enough that the confusables list does not contain any Arabic–Latin character pairs. Proposing a more nuanced definition of visual similarity for Arabic and the associated prior is left for future work.

We incorporate these mappings into the model as Dirichlet priors on the emission parameters: $\theta \sim \text{Dir}(\alpha)$, where each dimension of the parameter α corresponds to a character pair (c_o, c_l), and the corresponding element of α is set to the number of times these symbols are mapped to each other in the predefined mapping set.

3.3 Learning

We learn the emission WFST parameters in an unsupervised fashion, observing only the Latin side of the training instances. The marginal likelihood of a romanized sequence l can be computed by

[4] http://arabic.omaralzabir.com/, https://thomasplagwitz.com/2013/01/06/imrans-phonetic-keyboard-for-arabic/

[5] http://winrus.com/kbd_e.htm

[6] https://www.unicode.org/Public/security/latest/confusables.txt

[7] In our parameterization, we cannot introduce a mapping from one to multiple symbols or vice versa, so we map all possible pairs instead: (*ю*, *lo*) → (*ю*, *l*), (*ю*, *o*).

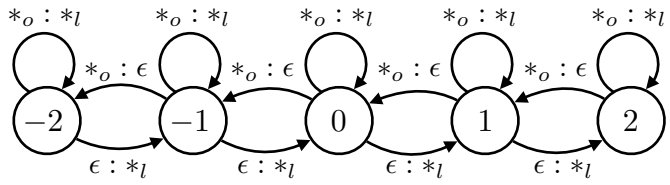

Figure 3: Schematic of the emission WFST with limited delay (here, up to 2) with states labeled by their delay values. $*_o$ and $*_l$ represent an arbitrary original or Latin symbol respectively. Weights of the arcs are omitted for clarity; weights with the same input-output label pairs are tied.

summing over the weights of all paths through a lattice obtained by composing $T \circ S \circ A(l)$. Here $A(l)$ is an unweighted acceptor of l, which, when composed with a lattice, constrains all paths through the lattice to produce l as the output sequence. The expectation–maximization (EM) algorithm is commonly used to maximize marginal likelihood; however, the size of the lattice would make the computation prohibitively slow. We combine online learning (Liang and Klein, 2009) and curriculum learning (Bengio et al., 2009) to achieve faster convergence, as described in §3.3.1.

3.3.1 Unsupervised learning

We use a version of the stepwise EM algorithm described by Liang and Klein (2009), reminiscent of the stochastic gradient descent in the space of the sufficient statistics. Training data is split into mini-batches, and after processing each mini-batch we update the overall vector of the sufficient statistics μ and re-estimate the parameters based on the updated vector. The update is performed by interpolating between the current value of the overall vector and the vector of sufficient statistics s_k collected from the k-th mini-batch: $\mu^{(k+1)} \leftarrow (1 - \eta_k)\mu^{(k)} + \eta_k s_k$. The stepsize is gradually decreased, causing the model to make smaller changes to the parameters as the learning stabilizes. Following Liang and Klein (2009), we set it to $\eta_k = (k + 2)^{-\beta}$.

However, if the mini-batch contains long sequences, summing over all paths in the corresponding lattices could still take a long time. As we know, the character substitutions are not arbitrary: each original alphabet symbols is likely to be mapped to only a few Latin characters, which means that most of the paths through the lattice would have very low probabilities. We prune the improbable arcs in the emission WFST while training on batches of shorter sentences. Doing this eliminates up to 66% and up to 76% of the emission arcs for Arabic and Russian respectively.

We discourage excessive use of insertions and deletions by keeping the corresponding probabilities low at the early stages of training: during the first several updates, we freeze the deletion probabilities at a small initial value and disable insertions completely to keep the model locally normalized. We also iteratively increase the language model order as learning progresses. Once most of the emission WFST arcs have been pruned, we can afford to compose it with a larger language model WFST without the size of the resulting lattice rendering the computation impractical. The two steps of the EM algorithm are performed as follows:

E-step At the E-step we compute the sufficient statistics for updating θ, which in our case would be the expected number of traversals of each of the emission WFST arcs. For ease of bookkeeping, we compute those expectations using finite-state methods in the expectation semiring (Eisner, 2002). Summing over all paths in the lattice is usually performed via shortest distance computation in log semiring; in the expectation semiring, we augment the weight of each arc with a basis vector, where the only non-zero element corresponds to the index of the emission edit operation associated with the arc (i.e. the input-output label pair). This way the shortest distance algorithm yields not only the marginal likelihood but also the vector of the sufficient statistics for the input sequence.

To speed up the shortest distance computation, we shrink the lattice by limiting delay of all paths through the emission WFST. Delay of a path is defined as the difference between the number of the epsilon labels on the input and output sides of the path. Figure 3 shows the schema of the emission WFST where delay is limited. Substitutions are performed without a state change, and each deletion or insertion arc transitions to the next or previous state respectively. When the first (last) state is reached, further deletions (insertions) are no longer allowed.

M-step The M-step then corresponds to simply re-estimating θ by appropriately normalizing the obtained expected counts.

	Arabic		Russian	
	Sent.	Char.	Sent.	Char.
LM train	49K	935K	307K	111M
Train	5K	104K	5K	319K
Validation	301	8K	227	15K
Test	1K	20K	1K	72K

Table 2: Splits of the Arabic and Russian data used in our experiments. All Arabic data comes from the LDC BOLT Phase 2 corpus, in which all sentences are annotated with their transliteration into the Arabic script. For the experiments on Russian, the language model is trained on a section of the Taiga corpus, and the train, validation, and test portions are collected by the authors; only the validation and test sentences are annotated.

3.3.2 Supervised learning

We also compare the performance of our model with the same model trained in a supervised way, using the annotated portion of the data that contains parallel o and l sequences. In the supervised case we can additionally constrain the lattice with an acceptor of the original orthography sequence: $A(o) \circ T \circ S \circ A(l)$. However, the alignment between the symbols in o and l is still latent. To optimize this marginal likelihood we still employ the EM algorithm. As this constrained lattice is much smaller, we can run the standard EM without the modifications discussed in §3.3.1.

3.4 Decoding

Inference at test time is also performed using finite-state methods and closely resembles the E-step of the unsupervised learning: given a Latin sequence l, we construct the machine $T \circ S \circ A(l)$ in the tropical semiring and run the shortest path algorithm to obtain the most probable path $\hat{e}$; the source sequence $\hat{o}$ is read off the obtained path.

4 Datasets

Here we discuss the data used to train the unsupervised model. Unlike Arabizi, which has been explored in prior work due to its popularity in the modern online community, a dataset of informally romanized Russian was not available, so we collect and partially annotate our own dataset from the Russian social network `vk.com`.

4.1 Arabic

We use the Arabizi portion of the LDC BOLT Phase 2 SMS/Chat dataset (Bies et al., 2014; Song et al., 2014), a collection of written informal conversations in romanized Egyptian Arabic annotated with their Arabic script representation. To prevent the annotators from introducing orthographic variation inherent to dialectal Arabic, compliance with the Conventional orthography for dialectal Arabic (CODA; Habash et al., 2012) is ensured. However, the effects of some of the normalization choices (e.g. expanding frequent abbreviations) would pose difficulties to our model.

To obtain a subset of the data better suited for our task, we discard any instances which are not originally romanized (5% of all data), ones where the Arabic annotation contains Latin characters (4%), or where emoji/emoticon normalization was performed (12%). The information about the splits is provided in Table 2. Most of the data is allocated to the language model training set in order to give the unsupervised model enough signal from the native script side. We choose to train the transition model on the annotations from the same corpus to make the language model specific to both the informal domain and the CODA orthography.

4.2 Russian

We collect our own dataset of romanized Russian text from a social network website `vk.com`, adopting an approach similar to the one described by Darwish (2014). We take a list of the 50 most frequent Russian lemmas (Lyashevskaya and Sharov, 2009), filtering out those shorter than 3 characters, and produce a set of candidate romanizations for each of them to use as queries to the `vk.com` API. In order to encourage diversity of romanization styles in our dataset, we generate the queries by defining all plausible visual and phonetic mappings for each Cyrillic character and applying all possible combinations of those substitutions to the underlying Russian word. We scrape public posts on the user and group pages, retaining only the information about which posts were authored by the same user, and manually go over the collected set to filter out coincidental results.

Our dataset consists of 1796 wall posts from 1681 users and communities. Since the posts are quite long on average (248 characters, longest ones up to 15K), we split them into sentences using the NLTK sentence tokenizer, with manual

correction when needed. The obtained sentences are used as data points, split into training, validation and test according to the numbers in Table 2. The average length of an obtained sentence is 65 characters, which is 3 times longer than an average Arabizi sentence; we believe this is due to the different nature of the data (social media posts vs. SMS). Sentences collected from the same user are distributed across different splits so that we observe a diverse set of romanization preferences in both training and testing. Each sentence in the validation and test sets is annotated by one of the two native speaker annotators, following guidelines similar to those designed for the Arabizi BOLT data (Bies et al., 2014). For more details on the annotation guidelines and inter-annotator agreement, see Appendix A.

Since we do not have enough annotations to train the Russian language model on the same corpus, we use a separate in-domain dataset. We take a portion of the Taiga dataset (Shavrina and Shapovalova, 2017), containing 307K comments scraped from the same social network `vk.com`, and apply the same preprocessing steps as we did in the collection process.

5 Experiments

Here we discuss the experimental setup used to determine how much information relevant for our task is contained in the character similarity mappings, and how it compares to the amount of information encoded in the human annotations. We compare them by evaluating the effect of the informative priors (described in §3.2) on the performance of the unsupervised model and comparing it to the performance of the supervised model.

Methods We compare the performance of our model trained in three different setups: unsupervised with a uniform prior on the emission parameters, unsupervised with informative phonetic and visual priors (§3.2), and supervised. We additionally compare them to a commercial online decoding system for each language (directly encoding human knowledge about the transliteration process) and a character-level unsupervised neural machine translation architecture (encoding no assumptions about the underlying process at all).

We train the unsupervised models with the stepwise EM algorithm as described in §3.3.1, performing stochastic updates and making only one pass over the entire training set. The supervised models are trained on the validation set with five iterations of EM with a six-gram transition model. It should be noted that only a subset of the validation data is actually used in the supervised training: if the absolute value of the delay of the emission WFST paths is limited by n, we will not be able to compose a lattice for any data points where the input and output sequences differ in length by more than n (those constitute 22% of the Arabic validation data and 33% of the Russian validation data for $n = 5$ and $n = 2$ respectively). Since all of the Arabic data comes annotated, we can perform the same experiment using the full training set; surprisingly, the performance of the supervised model does not improve (see Table 3).

The online transliteration decoding systems we use are `translit.net` for Russian and Yamli[8] for Arabic. The Russian decoder is rule-based, but the information about what algorithm the Arabic decoder uses is not disclosed.

We take the unsupervised neural machine translation (UNMT) model of Lample et al. (2018) as the neural baseline, using the implementation from the codebase of He et al. (2020), with one important difference: since the romanization process is known to be strictly character-level, we tokenize the text into characters rather than words.

Implementation We use the OpenFst library (Allauzen et al., 2007) for the implementation of all the finite-state methods, in conjunction with the OpenGrm NGram library (Roark et al., 2012) for training the transition model specifically. We train the character-level n-gram models with Witten–Bell smoothing (Witten and Bell, 1991) of orders from two to six. Since the WFSTs encoding full higher-order models become very large (for example, the Russian six-gram model has 3M states and 13M arcs), we shrink all the models except for the bigram one using relative entropy pruning (Stolcke, 1998). However, since pruning decreases the quality of the language model, we observe most of the improvement in accuracy while training with the unpruned bigram model, and the subsequent order increases lead to relatively minor gains. Hyperparameter settings for training the transition and emission WFSTs are described in Appendix B.

We optimize the delay limit for each language separately, obtaining best results with 2 for Russian and 5 for Arabic. To approximate the mono-

[8] `https://www.yamli.com/`

	Arabic	Russian
Unsupervised: uniform prior	0.735	0.660
Unsupervised: phonetic prior	0.377	0.222
Unsupervised: visual prior	—	0.372
Unsupervised: combined prior	—	0.212
Supervised	0.225*	0.140
UNMT	0.791	0.242
Commercial	0.206	0.137

Table 3: Character error rate for different experimental setups. We compare unsupervised models with and without informative priors with the supervised model (trained on validation data) and a commercial online system. We do not have a visual prior for Arabic due to the Arabic–Latin visual character similarity not being captured by the restrictive confusables list that defines the prior (see §3.2). Each supervised and unsupervised experiment is performed with 5 random restarts. *The Arabic supervised experiment result is for the model trained on the validation set; training on the 5K training set yields 0.226.

tonic word-level alignment between the original and Latin sequences, we restrict the operations on the space character to only three: insertion, deletion, and substitution with itself. We apply the same to the punctuation marks (with specialized substitutions for certain Arabic symbols, such as ? → ؟). This substantially reduces the number of arcs in the emission WFST, as punctuation marks make up over half of each of the alphabets.

Evaluation We use character error rate (CER) as our evaluation metric. We compute CER as the ratio of the character-level edit distance between the predicted original script sequence and the human annotation to the length of the annotation sequence in characters.

6 Results and analysis

The CER values for the models we compare are presented in Table 3. One trend we notice is that the error rate is lower for Russian than for Arabic in all the experiments, including the uniform prior setting, which suggests that decoding Arabizi is an inherently harder task. Some of the errors of the Arabic commercial system could be explained by the decoder predictions being plausible but not matching the CODA orthography of the reference.

Original	Latin
р /r/	*r* (.93), *p* (.05)
б /b/	*b* (.95), *6* (.02)
в /v/	*v* (.87), *8* (.05), *w* (.05)
و /w, uː, oː/	*w* (.48), *o* (.33), *u* (.06)
خ /x/	*5* (.76), *k* (.24)

Table 4: Emission probabilities learned by the supervised model (compare to Table 1). All substitutions with probability greater than 0.01 are shown.

Effect of priors The unsupervised models without an informative prior perform poorly for either language, which means that there is not enough signal in the language model alone under the training constraints we enforce. Possibly, the algorithm could have converged to a better local optimum if we did not use the online algorithm and prune both the language model and the emission model; however, that experiment would be infeasibly slow. Incorporating a phonetic prior reduces the error rate by 0.36 and 0.44 for Arabic and Russian respectively, which provides a substantial improvement while maintaining the efficiency advantage. The visual prior for Russian appears to be slightly less helpful, improving CER by 0.29. We attribute the better performance of the model with the phonetic prior to the sparsity and restrictiveness of the visually confusable symbol mappings, or it could be due to the phonetic substitutions being more popular with users. Finally, combining the two priors for Russian leads to a slight additional improvement in accuracy over the phonetic prior only.

We additionally verify that the phonetic and visual similarity-based substitutions are prominent in informal romanization by inspecting the emission parameters learned by the supervised model with a uniform prior (Table 4). We observe that: (a) the highest-probability substitutions can be explained by either phonetic or visual similarity, and (b) the external mappings we use for our priors are indeed appropriate since the supervised model recovers the same mappings in the annotated data.

Error analysis Figure 4 shows some of the elements of the confusion matrices for the test predictions of the best-performing unsupervised models in both languages. We see that many of the frequent errors are caused by the model failing to disambiguate between two plausible decodings of a Latin character, either mapped to it

through different types of similarity (*н* /n/ [phonetic] → *n* ← [visual] *n*, *н* [visual] → *h* ← [phonetic] *x* /x/), or the same one (visual 8 → 8 ← *в*, phonetic ه /h/ → *h* ← ح /ħ/); such cases could be ambiguous for humans to decode as well.

Other errors in Figure 4 illustrate the limitations of our parameterization and the resources we rely on. Our model does not allow one-to-many alignments, which leads to digraph interpretation errors such as س /s/ + ه /h/ → *sh* ← ش /ʃ/. Some artifacts of the resources our priors are based on also pollute the results: for example, the confusion between *ь* and *x* in Russian is explained by the Russian soft sign *ь*, which has no English phonetic equivalent, being arbitrarily mapped to the Latin *x* in one of the phonetic keyboard layouts.

Comparison to UNMT The unsupervised neural model trained on Russian performs only marginally worse than the unsupervised WFST model with an informative prior, demonstrating that with a sufficient amount of data the neural architecture is powerful enough to learn the character substitution rules without the need for the inductive bias. However, we cannot say the same about Arabic—with a smaller training set (see Table 2), the UNMT model is outperformed by the unsupervised WFST even without an informative prior. The main difference in the performance between the two models comes down to the trade-off between structure and power: although the neural architecture captures long-range dependencies better due to having a stronger language model, it does not provide an easy way of enforcing character-level constraints on the decoding process, which the WFST model encodes by design. As a result, we observe that while the UNMT model can recover whole words more successfully (for Russian it achieves 45.8 BLEU score, while the best-performing unsupervised WFST is at 20.4), it also tends to arbitrarily insert or repeat words in the output, which leads to higher CER.

7 Conclusion

This paper tackles the problem of decoding non-standardized informal romanization used in social media into the original orthography without parallel text. We train a WFST noisy-channel model to decode romanized Egyptian Arabic and Russian to their original scripts with the stepwise EM algorithm combined with curriculum learning and demonstrate that while the unsupervised model by itself performs poorly, introducing an informative prior that encodes the notion of phonetic or visual character similarity brings its performance substantially closer to that of the supervised model.

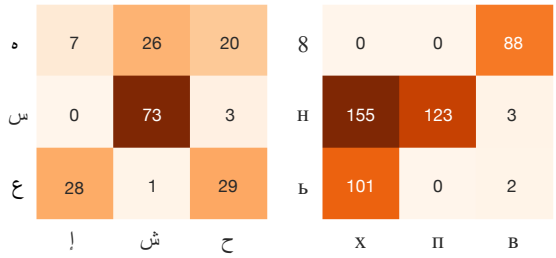

Figure 4: Fragments of the confusion matrix comparing test time predictions of the best-performing unsupervised models for Arabic (left) and Russian (right) to human annotations. Each number represents the count of the corresponding substitution in the best alignment (edit distance path) between the predicted and gold sequences, summed over the test set. Rows stand for predictions, columns correspond to ground truth.

The informative priors used in our experiments are constructed using sets of character mappings compiled for other purposes but using the same underlying principle (phonetic keyboard layouts and the Unicode confusable symbol list). While these mappings provide a convenient way to avoid formalizing the complex notions of the phonetic and visual similarity, they are restrictive and do not capture all the diverse aspects of similarity that idiosyncratic romanization uses, so designing more suitable priors via operationalizing the concept of character similarity could be a promising direction for future work. Another research avenue that could be explored is modeling specific user preferences: since each user likely favors a certain set of character substitutions, allowing user-specific parameters could improve decoding and be useful for authorship attribution.

Acknowledgments

This project is funded in part by the NSF under grants 1618044 and 1936155, and by the NEH under grant HAA256044-17. The authors thank John Wieting, Shruti Rijhwani, David Mortensen, Nikita Srivatsan, and Mahmoud Al Ismail for helpful discussion, Junxian He for help with the UNMT experiments, Stas Kashepava for data annotation, and the three anonymous reviewers for their valuable feedback.

References

Mohamed Al-Badrashiny, Ramy Eskander, Nizar Habash, and Owen Rambow. 2014. Automatic transliteration of Romanized dialectal Arabic. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 30–38, Ann Arbor, Michigan. Association for Computational Linguistics.

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 40–47, Sapporo, Japan. 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.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

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.

Aimilios Chalamandaris, Athanassios Protopapas, Pirros Tsiakoulis, and Spyros Raptis. 2006. All Greek to me! An automatic Greeklish to Greek transliteration system. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Aimilios Chalamandaris, Pirros Tsiakoulis, Spyros Raptis, G Giannopoulos, and George Carayannis. 2004. Bypassing Greeklish! In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

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.

Kareem Darwish, Walid Magdy, and Ahmed Mourad. 2012. Language processing for arabic microblog retrieval. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 2427–2430, New York, NY, USA. Association for Computing Machinery.

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.

Ramy Eskander, Mohamed Al-Badrashiny, Nizar Habash, and Owen Rambow. 2014. Foreign words and the automatic processing of Arabic social media text written in Roman script. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 1–12, Doha, Qatar. Association for Computational Linguistics.

Evgeniy Gabrilovich and Alex Gontmakher. 2002. The homograph attack. *Commun. ACM*, 45(2):128.

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).

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*.

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.

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

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.

Olga N Lyashevskaya and Sergey A Sharov. 2009. *Frequency dictionary of modern Russian based on the Russian National Corpus [Chastotnyy slovar' sovremennogo russkogo jazyka (na materiale Nacional'nogo korpusa russkogo jazyka)]*. Azbukovnik, Moscow.

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.

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.

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).

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270—274.

Ian H Witten and Timothy C Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE transactions on information theory*, 37(4):1085–1094.

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.

A Data collection and annotation

Preprocessing We generate a set of 270 candidate transliterations of 26 Russian words to use as queries. However, many of the produced combinations are highly unlikely and yield no results, and some happen to share the spelling with words in other languages (most often other Slavic languages that use Latin script, such as Polish). We scrape public posts on user and group pages, retaining only the information about which posts were authored by the same user, and manually go over the collected set to filter out coincidental results. We additionally preprocess the collected data by normalizing punctuation and removing non-ASCII characters and emoji. We also replace all substrings of the same character repeated more than twice to only two repetitions, as suggested by Darwish et al. (2012), since these repetitions are more likely to be a written expression of emotion than to be explained by the underlying Russian sentence. The same preprocessing steps are applied to the original script side of the data (the annotations and the monolingual language model training corpus) as well.

Annotation guidelines While transliterating, annotators perform orthographic normalization wherever possible, correcting typos and errors in word boundaries; grammatical errors are not corrected. Tokens that do not require transliteration (foreign words, emoticons) or ones that annotator fails to identify (proper names, badly misspelled words) are removed from the romanized sentence and not transliterated. Although it means that some of the test set sentences will not exactly represent the original romanized sequence, it will help us ensure that we are only testing our model's ability to transliterate rather than make word-by-word normalization decisions.

In addition, 200 of the validation sequences are dually annotated to measure the inter-annotator agreement. We evaluate it using character error rate (CER; edit distance between the two sequences normalized by the length of the reference sequence), the same metric we use to evaluate the model's performance. In this case, since neither of the annotations is the ground truth, we compute CER in both directions and average. Despite the discrepancies caused by the annotators deleting unknown words at their discretion, average CER is only 0.014, which indicates a very high level of agreement.

B Hyperparameter settings

WFST model The Witten–Bell smoothing parameter for the language model is set to 10, and the relative entropy pruning threshold is 10^{-5} for the trigram model and $2 \cdot 10^{-5}$ for higher-order models. Unsupervised training is performed in batches of size 10 and the language model order is increased every 100 batches. While training with the bigram model, we disallow insertions and freeze all the deletion probabilities at e^{-100}. The EM stepsize decay rate is $\beta = 0.9$. The emission arc pruning threshold is gradually decreased from 5 to 4.5 (in the negative log probability space). We perform multiple random restarts for each experiment, initializing the emission distribution to uniform plus random noise.

UNMT baseline Our unsupervised neural baseline uses a single-layer LSTM with hidden state size 512 for both the encoder and the decoder. The embedding dimension is set to 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 tune the maximum training sequence length (controlling how much training data is used) and the maximum allowed decoding length by optimizing the validation set CER. In our case, the maximum output length is important because the evaluation metric penalizes the discrepancy in length between the prediction and the reference; we observe the best results when setting it to 40 characters for Arabic and 180 for Russian. At training time, we filter out sequences longer than 100 characters for either language, which constitute 1% of the available Arabic training data (both the Arabic-only LM training set and the Latin-only training set combined) but almost 70% of the Russian data. Surprisingly, the Russian model trained on the remaining 30% achieves better results than the one trained on the full data; we hypothesize that the improvement comes from having a more balanced training set, since the full data is heavily skewed towards the Cyrillic side (LM training set) otherwise (see Table 2).