2020. *Proceedings of the Workshop on Turkic and Languages in Contact with Turkic* 5. 173–185.
https://doi.org/10.3765/ptu.v5i1.4783

Multi-script morphological transducers and transcribers for seven Turkic languages

Jonathan N. Washington & Francis M. Tyers & Oğuzhan Kuyrukçu *

Abstract. This paper describes ongoing work to augment morphological transducers for seven Turkic languages with support for multiple scripts each, as well as preliminary work adding IPA transcription systems. Evaluation demonstrates that our approach yields coverage equivalent to or not much lower than that of the base transducers.

Keywords. morphological transducers; orthography; Turkic languages

1. Introduction. Of the existing Free/Open-Source morphological transducers for Turkic languages (Washington et al. 2020), each is implemented in only one orthography, despite a number of the languages being currently written in two or more orthographies, or having a large body of text written in an orthography that has been in use recently.

A morphological transducer converts between form and analysis—for example, `алмалардан` ↔ `алма<n><pl><abl>`—where the form-to-analysis task is termed "morphological analysis" and the analysis-to-form task is termed "morphological generation".

This paper builds on work in which Cyrillic support was added to a transducer for Crimean Tatar which had been implemented in the Latin script (Tyers et al. 2019). We leverage morphological transducers for Kazakh (implemented in the Cyrillic script), Kyrgyz (Cyrillic), Turkmen (Latin), Qaraqalpaq (Latin), Uzbek (Latin), and Uyghur (Perso-Arabic), and add support for analysis and generation in additional scripts that are currently or have recently been used for the languages. Specifically, we add Cyrillic support to Turkmen, Qaraqalpaq, Uzbek, and Uyghur transducers; Perso-Arabic support to the Kazakh and Kyrgyz transducers; Latin script to the Kazakh and Uyghur transducers; and additional Latin orthographies to the Qaraqalpaq transducer. The addition of International Phonetic Alphabet (IPA) transcription support to these transducers has also begun, and initial work on adding such support to the Kyrgyz transducer is discussed.

Motivation for this work is plentiful. Support for multiple scripts allows academic work that employs corpora of these languages to expand to include text from more sources.[1] More importantly, language technology that uses these transducers as a component, such as machine translation systems and spell checkers, will now be accessible to more communities that use the languages. Additionally, these systems may be used to simply convert between the orthographies, a task that has the potential to enable textual exchange between different communities using a single language with different scripts. We further expect the IPA transcribers to be useful in the development of text-to-speech systems for these languages. By providing these resources publicly under free/open source licenses,[2] we hope that others may use and build on our work.

* Thank you to the audience of Tu+5, two anonymous reviewers of our abstract, and two anonymous reviewers of this paper. We also very much appreciate the contributions of Connor Mayer to the Uyghur transliteration transducer since our presentation. This article contains the output of a research project implemented as part of the Basic Research Program at the National Research University Higher School of Economics (HSE University). Authors: Jonathan Washington, Swarthmore College (jonathan.washington@swarthmore.edu), Francis Tyers, Indiana University and Высшая Школа Экономики (ftyers@iu.edu), Oğuzhan Kuyrukçu, Boğaziçi Üniversitesi (kuyrukcuoguz@gmail.com).

[1] For example, Mayer, Major & Yakup (n.d.) have begun to leverage the Uyghur transducers described in this paper to study vowel-harmony transparent vowels in Uyghur as written in different scripts.

[2] All code underlying the tools presented here is available in the respective language repositories at http://github.com/apertium. The code may be demoed at http://turkic.apertium.org/

Our particular approach automates the construction of new transducers: an analyser for each language that supports multiple orthographies, and individual generators for each orthography, all with a uniform interface (analysis side) across orthographies. This is preferable to introducing an additional tool for transcription for several reasons: it maintains an efficient pipeline with a minimum number of tools and it allows for the resulting transducers to be used as drop-in replacements for the current transducers in any of the contexts they're currently used in (spell checking, machine translation) or might be used in in the future. Furthermore, it leverages the strengths of finite-state technology: errors can be easily adjusted to improve accuracy. Corpus-based (neural or statistical) approaches to transliteration, such as those used by Saini & Lehal (2008), Bao et al. (2013), and Liu et al. (2018), do not offer these advantages, and we are not aware of any non-automatically-generated parallel corpora of these languages that such models might be trained on.

1.1. Morphological Transducers. A morphological transducer is a finite-state transducer (FST) that maps between linguistic forms and morphological analyses. Examples of morphological analysis and generation are shown in Table 1.

Table 1: Morphological analysis and generation of four forms of the word for 'apple' (алма [ɑɬmɑ]) in several Turkic languages. These forms happen to be orthographically correct (and cognate) in the Cyrillic orthographies for most of the languages described in this paper (all except Uyghur and Uzbek). All forms have a part-of-speech tag <n> (noun); the remaining tags are grammatical tags: <nom> (nominative case), <abl> (ablative case), <pl> (plural).

(a) morphological analysis

input	output
алма	алма<n><nom>
алмадан	алма<n><abl>
алмалар	алма<n><pl><nom>
алмалардан	алма<n><pl><abl>

(b) morphological generation

input	output
алма<n><nom>	алма
алма<n><abl>	алмадан
алма<n><pl><nom>	алмалар
алма<n><pl><abl>	алмалардан

An FST is implemented as a state machine with states connected by labelled arcs. Each arc label has two sides—a "form" side and an "analysis" side. An FST processor matches input to a specified side of the arcs, and outputs the accumulated contents of the other side only when the end of the input coincides with an end state in the graph. A graph matching the forms in Table 1 is depicted in Figure 1.

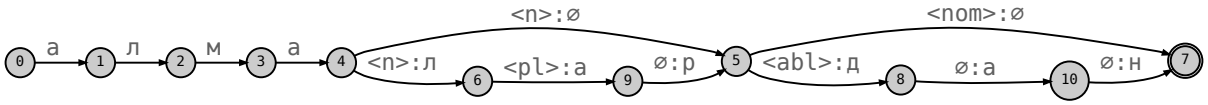


Figure 1: A finite-state transducer that implements the form-analysis mappings in Table 1. The : symbol separates the analysis and form sides (respectively) of the arc labels (when the two sides are different). State 0 is the start state, and a double circle (here only on state 7) indicates that a state is an end-state. Reading arc labels from left to right (across the :) performs morphological generation, and reading arc labels from right to left performs morphological analysis.

While there are other approaches to implementing morphological analysers and generators for a given language, FSTs offer an approach where analysis and generation of a given form can be accomplished by a single implementation. Additionally, FSTs are useful for a range of morphological

processing applications (Lindén et al. 2013), including corpus searching (e.g., as mentioned in footnote 1), machine translation (Forcada et al. 2011), spell-checking (Pirinen & Lindén 2014), predictive text (Forcada 2001, Silfverberg, Hyvärinen & Pirinen 2011), electronic dictionaries (Johnson, Antonsen & Trosterud 2013), and computer-aided language learning (Katinskaia, Nouri & Yangarber 2018).

1.2. Turkic transducers. Washington et al. (2020) present Free/Open Source morphological transducers for around 20 Turkic languages. These transducers were developed for use in machine translation systems, for use as spellcheckers, and for other language technology purposes. Their FOS licenses allow them to be studied, modified, and redistributed freely, for any purpose.

The transducers are hand-crafted, meaning that they are not "learned" using statistics over a corpus, but directly implement the morphology and lexicon of a language. Washington et al. (2016) and Tyers, Pirinen & Washington (2015) document the development process. There are several advantages to this approach, the main ones being accuracy and completeness. Because of the level of control over the morphology, an FST can be made to not undergenerate and not overgenerate forms. For example, every stem that should gets complete morphology—even forms not seen in a corpus—and morphemes are not combined improperly, such as with stems of the wrong part of speech. These are problems afflicting other approaches to morphological analysis and generation, such as neural (Kann, Cotterell & Schütze 2016, Silfverberg & Tyers 2019), affix-stripping/stemming (Eryiğit & Adalı 2004, Bimba et al. 2016), and various ad hoc approaches (Dovudov & Baisa 2010, Dovudov, Suchomel & Šmerk 2012).

Most of the transducers included in Washington et al. (2020) support nearly all the productive morphology in the languages they are implemented for, and have large lexicons (around or greater than 10K entries). The transducers are categorised by development status: production-level, working, and prototype transducers. Production-level transducers have near perfect coverage of the languages' morphology (including morphotactics and morphophonology), and a fairly comprehensive lexicon. "Working" transducers have shortcomings in at least one of these areas, though not ones that lower their accuracy immensely. Prototype transducers have major shortcomings in at least one of these areas which lower their accuracy significantly.

The production-level transducers presented by Washington et al. (2020) are for Tatar, Kazakh, Turkish, Kyrgyz, Crimean Tatar, and Tuvan, and in terms of naïve coverage, return analyses for 92% to 98% of forms in corpora. The working transducers are for Bashqort, Chuvash, Uzbek, Qaraqalpaq, Uyghur, Sakha, Qarachay, Gagauz, and Qumuq, and have naïve coverage ranges of 88% to 93%. The prototype transducers are for Azerbaycani, Turkmen, Noghay, Khakas, Altay, and Old Turkic, and have naïve coverage ranges of under 80%. Figure 2 shows the relative sizes of the lexicons of all of these transducers divided by word category (part of speech).

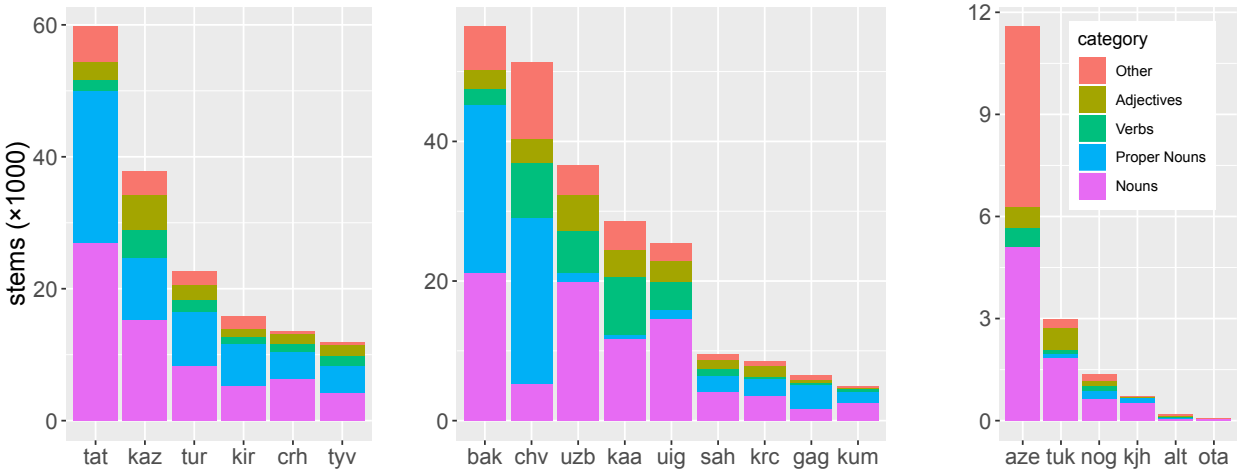
1.3. Turkic orthographies. This paper focusses on seven Turkic languages with FOS morphological transducers and for which recent textual resources exist in more than one orthography. Crimean Tatar, Kazakh, Kyrgyz, Qaraqalpaq, Turkmen, Uyghur, and Uzbek all commonly written in—or have recently been commonly written in—at least two of Cyrillic, the Latin script, and the Perso-Arabic script. These uses are summarised in Table 2.

There are several potential causes for multiple scripts to be in use for a given language:

- **different uses across national borders: Kyrgyz**, **Kazakh**, and **Uyghur** are written primarily in the Perso-Arabic script in China, but in Cyrillic in Kazakhstan and Kyrgyzstan;[3] and Turkmen and Uzbek are written in Perso-Arabic script in Afghanistan and/or Iran[4] but in Latin script

[3] In the case of Uyghur, there is no official policy for writing it outside of China, but Uyghur-speaking communities in the former Soviet Union write the language using Cyrillic.

[4] Both Turkmen and Uzbek are recognised as regional languages in Afghanistan, but Turkmen, while spoken in Iran,



(a) Production-level (b) Working (c) Prototype

Figure 2: Lexicon sizes (number of stems) for Turkic FSTs presented in Washington et al. (2020), broken down by word category. The languages are divided into three groups by FST quality.

Table 2: The scripts in current and recent use to write the languages discussed in this paper. Lack of a grey background (and in the case of Qaraqalpaq, the addition of an underline) indicates the script that the FOS transducer for that language is primarily implemented in, and — indicates that a script has not been in formalised common use for the language recently.

	Perso-Arabic	Cyrillic	Latin
Crimean Tatar	—	**official** in Republic of Crimea	widely used since 1990s
Kazakh	**official** in China	(still) **official** in Kazakhstan	(soon to be) **official** in Kazakhstan
Kyrgyz	**official** in China	**official** in Kyrgyzstan	—
Qaraqalpaq	—	previously **official**	≥3 recent **official** versions (1994, 2006, 2016)
Turkmen	used in Iran and Afghanistan	previously **official**, still widespread	**official** in Turkmenistan
Uyghur	**official** in China	previously **official** in Soviet Union	used in diaspora
Uzbek	used in Afghanistan	previously **official**, still widespread	**official** in Uzbekistan

in the former Soviet Union (Turkmenistan and Uzbekistan, respectively).

- **changes in policy:** There is an upcoming official switch from Cyrillic to Latin script for **Kazakh** in Kazakhstan, and a corresponding switch for **Qaraqalpaq**, **Turkmen**, and **Uzbek** has occurred in Uzbekistan and Turkmenistan within recent decades.
- **widely used but unrecognised standards:** The Latin script has been in wide use for Crimean Tatar in Crimea since the 1990s, and the Latin script is used for Uyghur in diaspora.

These languages all have at least half a million speakers each, but some are experiencing more disruption of cultural transmission than others. Until the collapse of the Soviet Union, all seven languages could be termed "stateless"; now Kazakh, Kyrgyz, Uzbek, and Turkmen all have state support at the national level in one country each, but the remainder do not. All seven languages are spoken by communities in countries where there is no state sponsorship of the language, and where the languages' speakers are minoritised. These communities are most often the ones using an orthography other than one the transducers were developed in, so making resources in a more widely used script accessible to these communities has the potential to increase access to the digital world in their languages. Kornai (2013) argues that access to digital resources in a language is now a prerequisite to that language's survival into future generations.

2. Methodology. Adding an orthography to a transducer should have the end result of enabling analysis (forms to analyses) of the new orthography in the new transducer, and allowing generation (analyses to forms) in the new orthography. Since we are dealing with tools that only allow two-tape transducers (that is, an analysis side and a form side for each arc label, with no additional levels of mapping), this means we need to create two equivalent generation transducers, but we may keep a single analysis transducer. As an example, the transducer that operates as shown in Table 1, with Cyrillic analysis and Cyrillic generation, will operate as shown in Table 3 after the addition of another script. In this example, Latin script is added, so now the analysis transducer will analyse Cyrillic and Latin script, and there are two separate but equivalent generation transducers: one for Cyrillic and one for Latin script.

Table 3: The desired performance of a morphological analyser and morphological generators with the addition of Latin script to the transducer modeled in table 1.

(a) bi-scriptual morphological analysis

input	output
алма	алма<n><nom>
alma	алма<n><nom>
алмадан	алма<n><abl>
almadan	алма<n><abl>
алмалар	алма<n><pl><nom>
almalar	алма<n><pl><nom>
алмалардан	алма<n><pl><abl>
almalardan	алма<n><pl><abl>

(b) Cyrillic morphological generation

input	output
алма<n><nom>	алма
алма<n><abl>	алмадан
алма<n><pl><nom>	алмалар
алма<n><pl><abl>	алмалардан

(c) Latin script morphological generation

input	output
алма<n><nom>	alma
алма<n><abl>	almadan
алма<n><pl><nom>	almalar
алма<n><pl><abl>	almalardan

has no official status or recognition there.

In keeping the analysis side of the transducer in the original script, we preserve the generator's interface to other tools, such as machine translation. Hypothetically the analysis side of the transducers could be converted if a particular need for it were identified.

Our approach employs HFST (Linden et al. 2011) tools to create some new FSTs that model orthographic conversion, and combine them with the existing transducer. We first create a transliteration transducer, consisting of two transducers compose-intersected:[5]

1. a lexical transducer (in `lexc` format) which maps all possible combinations of all characters used in the language, and
2. a two-level phonology FST (in `twol` format), where the "phonology" consists of character mappings sensitive to context.

This transliteration transducer (minimal example shown in Figure 3) is then compose-intersected with the existing transducer (Figure 1) to create a transducer in the new script, as shown in Figure 4. The resulting transducer may be used as is for generation of new-script forms, and is unioned with the original transducer to create a combined analyser for both scripts, as shown in Figure 5. The steps are repeated for a third script to create tri-scriptual transducers, and so on. All of the steps needed to produce these outcomes are in the Makefiles for the transducers in their respective repositories.

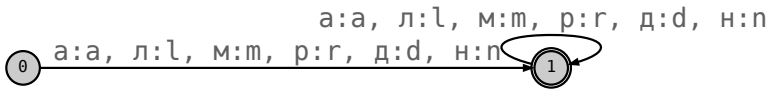


Figure 3: A Cyrillic-to-Latin script transliteration transducer that only handles the 6 characters in the example.

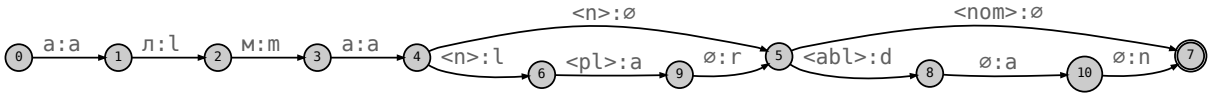


Figure 4: Latin-script FST, the result of compose-intersecting the transducer in Figure 1 with the transliteration transducer in Figure 3. Used for Latin-script generation.

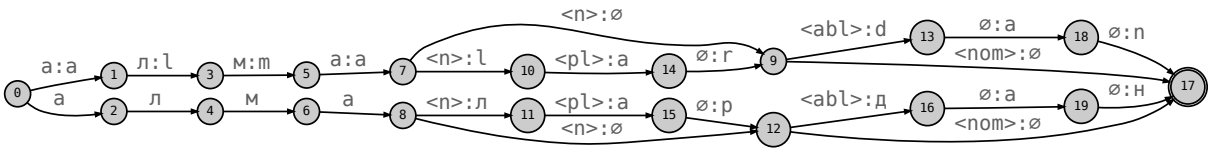


Figure 5: Biscriptual (Cyrillic and Latin script) FST, the result of unioning the Cyrillic transducer in Figure 1 with the Latin transducer in Figure 4. Used for multiscript analysis generation.

3. Challenges. This section presents a selection of the challenges encountered, and describes their solutions, or what would be needed to fully solve them. These challenges also serve as further examples of the methodology.

[5] Compose-intersect is an operation whereby the result is equivalent to the composition of all the `twol` transducers in the first argument intersected with another transducer in the the second argument, but in a less serialised process.

3.1. 1-LETTER TO 2-LETTER MAPPINGS. One common problem in mapping between orthographies is the case of one-to-many mappings; fortunately, this challenge is easily solved under the current approach.

As an example, many of the Cyrillic orthographies we have dealt with have letters that represent the glide /j/ followed by a vowel, whereas the Latin and Perso-Arabic orthographies write /j/ and a vowel as two separate characters. For example, Uzbek Cyrillic ‹я› corresponds to Uzbek Latin script ‹ya›.

If this were implemented entirely in the phonological transducer, it would require always converting relevant Cyrillic letters to the plain vowel equivalent in the target script and inserting a /j/ letter in the target script immediately preceding that change. This exposes a complication of using `twol` transducers for the entire transliteration transducer: insertion of an arc where there was not one, while possible, can lead to unexpected or unpredictable results.

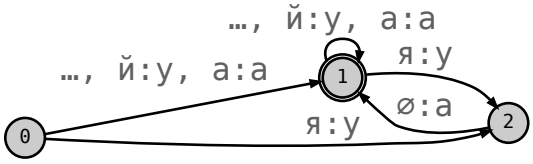
One of the easier ways to solve this problem is to map each /jV/ Cyrillic letter to two Cyrillic letters in the lexical transducer, e.g. `я:йа`. Then the phonological transducer may simply match the individual characters: `й:y` and `a:a`. (The final result in this example is `я:ya`.) Relevant excerpts from the lexical transducer code (`lexc` file) and phonological transducer code (`twol` file) for this example and the relevant excerpt from the resulting transliteration transducer are shown in Figure 6.

```
я:йа
й:й
a:a
…:…
```

(a) The `lexc` code (first level)

```
й:y
a:a
…:…
```

(b) The `twol` code (second level)



(c) The FST created by the code snippits.

Figure 6: Code snippits and an excerpt of an FST which can be used to convert the character ‹я› to two characters ‹ya› (and may also operate in the reverse direction).

Another example of this sort of mapping can been seen in the mapping of pre-2016 Latin-script Qaraqalpaq to post-2016 Latin-script Qaraqalpaq. The earlier system used an apostrophe diacritic for some characters that now are written with a combining diacritic, so that e.g. the following mappings are needed: `ń:n'`, `ǵ:g'`, `á:a'`, `ó:o'`, and `ú:u'`.

The reverse situation, where two characters need to be converted to a single character, can be solved with `twol` rules. However, it is potentially more complicated if the two characters are sometimes transliterated as single separate letters. An example is Uzbek Latin script ‹sh›, which can correspond to Cyrillic ‹ш› (as in bosh/бош ‘head’) or ‹сҳ› (as in Isҳoq/Исҳоқ ‘Isaac’). This lack of determinism can be addressed using the methodology outlined in section 3.6.

3.2. KAZAKH DAYEKSHE. Another potential challenge is seen in dealing with Kazakh *dayekshe*. The *dayekshe* is a diacritic that is written at the beginning of front-vowel words in Kazakh in which an unambiguously “front” letter is not present.

For example, a word like ‹تىلى› [tɘlɘ] ‘language–POSS.3’ is written with an initial *dayekshe* to show that it doesn’t represent a back-vowel word pronounced *[tɘɬɘ]. However ‹تىلىندە› [tɘlɘndiɘ] ‘language–POSS.3–LOC’ is not written with a *dayekshe* because ‹ە› [iɘ] is an unambiguous front-vowel character, and similarly ‹تىلىك› [tɘlɘk] ‘thin slice’ is not written with a *dayekshe* because ‹ك› [k] is an unambiguous front letter (although not a vowel, it almost never occurs adjacent to back vowels in the “native” lexicon).

The correct placement of *dayekshe* is solved by first inserting a special symbol at the beginning of all words in the lexical transducer (`lexc`). In the phonological transducer (`twol`), the character is deleted by default, and restricted to output as a *dayekshe* only in words where a Cyrillic front-vowel

letter that is ambiguous in backness in the Perso-Arabic script is used, and no unambiguously front character is in the word.

There are four letters in the Kazakh Perso-Arabic script which are ambiguous in backness with regards to their correspondence with the Cyrillic script: ‹ا› corresponds to either ‹а› [ɑ] or ‹ә› [æ]; ‹و› corresponds to either ‹о› [uʊ] or ‹ө› [yʉ], ‹ۇ› corresponds to either ‹ұ› [ʊ] or ‹ү› [ʉ], and ‹ى› corresponds to either ‹ы› [ə] or ‹і› [ɘ]. Each of the eight corresponding Cyrillic letters are hence simply mapped to the proper Perso-Arabic script letter, and no further attention is needed.

3.3. Spelling variants. Another common problem is that some characters are represented multiple ways depending on where the text is taken from.

For example, the apostrophe-like diacritic in pre-2016 Latin-script Qaraqalpaq may be represented as ⟨'⟩ apostrophe (U+0027), ⟨ʻ⟩ modifier letter turned comma / ʻokina (U+02BB), ⟨ʼ⟩ modifier letter apostrophe (U+02BC), ⟨‘⟩ opening single quote (U+2018), ⟨’⟩ closing single quote (U+2019), and ⟨`⟩ backtick (U+0060), among others. Another example is the Kazakh *dayekshe*, discussed above, which is alternatingly represented as ⟨ء⟩ hamza (U+0621) hamza or ⟨ٔ⟩ high hamza (U+0674).

For situations like these, we implement an additional two-level phonology transducer that maps each of the variants to a chosen "standard" which we use in the base transducer. This `twol` transducer is compose-intersected with the analyser transducer so that multiple variants may be recognised by the resulting analysis transducer. The generation transducer maintains the "standard" form in the base transducer. Examples of this solution have been implemented with the base transducers for a number of years (referred to as "spell relax"), but had not yet been documented in print.

3.4. Stress marks. Two different challenges encountered in transcription transducers are the inclusion of stress marks in the correct place and for their placement to be morphologically conditioned.

Stress marks in the International Phonetic Alphabet (IPA) precede the syllable they apply to, as opposed to the vowel, so there is often a consonant intervening between a stress mark and the vowel it might be seen as associated with; for example, [ɑɬˈmɑ] 'apple' is stressed on the second syllable, which the stress mark is placed at the beginning of—in this case, between two consonants. However, the identification of syllables is by their nuclei—that is, vowels. Identifying the edges of syllables is somewhat more complicated. Furthermore, even carefully crafted `twol` rules which limit the mapping between an empty character and a non-empty character (e.g., the `0:'` correspondence here) can be unreliable.

Our approach to placing stress marks in the appropriate place according to IPA convention is to use the `lexc` transducer component of the transliteration transducer (as described in section 2) to insert a "place-holder" symbol for a stress mark before every alphabetic character (resulting in an intermediate stage with forms like `{'}a{'}л{'}м{'}a`), and then crafting a deletion rule in the `twol` transducer to limit the realisation of those characters as a stress mark to only environments determined to be the beginning of a stressed syllable (with an empty realisation elsewhere). Writing such `twol` rules, with proper conditioning so as to correctly match the beginning of a syllable, is a challenge. Preliminary testing, however, shows that we have this essentially working in the Kyrgyz transcriber.

Another problem for proper treatment of stress is morphological and lexical "exceptions" to what is otherwise simply word-final stress in the Turkic languages discussed here. At the time of composition with the transcriber transducer, morphological and lexical information is not available. One solution would be to add stress information to the base transducers that is normally hidden, but could be made available to the transducers that determine which extraneous stress marks should be removed. Another solution would be to have three-level transducers (at one stage), where stress (and other transcription) rules can be conditioned on the contents of two other levels (morphological and orthographic); unfortunately, though, the `twol` compiler is not currently able to deal with more than two transducer levels. We have not yet attempted either of these solutions.

3.5. Competing mappings: simple cases. A more complicated set of challenges arise when there are multiple possibilities for converting a single character to another script. For example, the Kyrgyz Cyrillic script, in which the Kyrgyz base transducer is designed, represents both /q/ and /k/ with the same character ‹к›, but the Kyrgyz Perso-Arabic script uses two separate characters: ‹ك› and ‹ق›. In this case, the choice can usually be conditioned on surrounding vowels. Some exceptions where the surrounding vowels offer competing cues—usually compounds or borrowings, such as Бегайым:بەگايىم ‘Begayim (female name)’ and багелек:باعەلەك ‘pant bottoms’—must be included explicitly in the list of environments either where a rule does or does not apply. These Kyrgyz-specific issues are also dealt with in the IPA transliteration transducer for Kyrgyz, and this type of problem is particularly common for IPA transcription transducers.

3.6. Competing mappings: Crimean Tatar Cyrillic. A somewhat more complicated version of this problem is generating Russian words correctly in Russian orthography from Crimean Tatar Latin orthography, which does not have equivalents for Cyrillic characters found only in Russian borrowings, such as hard sign and soft sign—e.g., ‹yanvar› ‘January’ as ‹январь› instead of *‹январ›. To increase the level of determinism in the conversion, an *n*-gram language model is used to weight the Cyrillic side of the end transducer, essentially biasing it towards Russian spellings.

Another issue in Crimean Tatar is that there are multiple mappings for front rounded vowels. For example, two words spelled the same in Crimean Tatar Latin script orthography (and pronounced the same) are ‹kör› ‘blind’, and ‹kör› ‘see’; in Cyrillic, these are spelled ‹кёр› and ‹корь›, respectively. Similarly, ‹yüz› ‘hundred’ and ‹yüz› ‘face’ are spelled ‹юз› and ‹юзь›, respectively, while ‹yut› ‘swallow’ is spelled ‹ют›. Much of this mess can be dealt with by `twol` rules, but not all of it (especially the homophonous examples). The methodology mentioned above begins to solve this problem as well—i.e., a weighted transducer based on *n*-gram frequencies from a corpus is compose-intersected with the unaugmented transliteration transducer. However, it is also limited in that there will be ambiguous forms that cannot be generated correctly.

Further details on this approach are discussed in Tyers et al. (2019).

3.7. Competing mappings: Uyghur Cyrillic. Another variant of this problem is correctly generating forms with correct letter casing when the original transducer is implemented in the Perso-Arabic script (which does not have letter case). This is a problem for adding Cyrillic support to the Uyghur transducer, where the Cyrillic orthography for Cyrillic uses capital letters at the beginning of proper names and sentence-initially. For example, ‹قازاقستان› ‘Kazakhstan’ is ‹Қазақстан›, with an initial uppercase letter, while ‹قازاقستانلىق› ‘Kazakhstani’ ‹қазақстанлық› is written in Cyrillic with an initial lowercase letter; likewise, ‹تىل› ‘language’ can be written in Cyrillic as ‹Тил› with an uppercase initial letter or ‹тил› with a lowercase initial letter depending on whether it is at the beginning of a sentence or not (respectively).

For now we are able to analyse almost all correct forms in this regard by having the transcriber produce multiple versions of each word: one all lowercase, one all uppercase, and one with an initial uppercase letter followed by all lowercase letters. This does not solve the generation problem, though.

One potential solution to this challenge might involve an *n*-gram language model, similarly to how the non-determinism in Latin-to-Cyrillic conversion for Crimean Tatar is currently dealt with.

4. Evaluation. We report two forms of evaluation for these multi-scriptual transducers. Section 4.1 presents the results of naïve coverage, of the number of forms in a corpus for which an analysis is returned, whether correct or not, over equivalent corpora for multiple scripts. Section 4.2 presents a more in-depth analysis of the reasons for which forms in the corpus did not receive analyses.

4.1. Naïve coverage. Naïve coverage is measured as the percentage of forms in a corpus for which an analysis is returned, whether correct or not.

We measured each transducer on a partial or full Bible translation. Because of the different sources for different scripts, the translations in a single language were not identical, but represented distinct texts—with at least different word choices and phrasings, but in most cases likely translated from entirely different source versions and languages. In a few cases, a Wikipedia dump (for Latin-script Crimean Tatar) or a Quran translation (Cyrillic Uzbek) were used. However, all corpora used to evaluate the transducers ranged in size from about 200,000 forms to about 700,000 forms.

Table 4: Naïve coverage of the analysers. Grey cells indicate the orthography that the base transducer is implemented in (a baseline for comparison), and — indicates that an orthography is not used or has not [yet] been implemented.

	Perso-Arabic	Cyrillic	Latin
Crimean Tatar	—	91.27%	91.93%
Kazakh	94.62%	96.53%	—
Kyrgyz	93.59%	93.37%	—
Qaraqalpaq	—	92.94%	91.84%, [no corpus]
Turkmen	—	70.64%	69.00%
Uyghur	90.54%	90.69%	—
Uzbek	—	91.30%	94.40%

The results of our naïve evaluation are presented in Table 4. The versions of the codebases evaluated are from mid May, 2020. As seen in the results, most of the transliterated analysers have coverage results that are not meaningfully different from those of the base transducer. The ones that are enough lower to suggest potential shortcomings in translation are Perso-Arabic Kazakh and Cyrillic Uzbek.[6] Qaraqalpaq is difficult to know, since the transducer is currently based on the 2016 Latin orthography—with transliterators for pre-2016 Latin orthographies (supporting pre- and post-2009 revisions) and the Cyrillic orthography—but large corpora are not available in the 2016 Latin orthography.

Similarly, we are unable to evaluate coverage on our IPA transcribers without first creating a gold standard from a list of random forms. However, none of the IPA transcribers we are developing are mature enough for us to consider it a good idea to create a test set.

4.2. Qualitative evaluation. In our qualitative evaluation, we chose 1200 random forms from a corpus (the same as the ones used for coverage), and evaluate the cause of any unanalysed forms for transducers that were built with transliterators (i.e., all but the base transducers). To date, we have only conducted a qualitative evaluation of two transducers.

Broadly, we identify two types of errors—(1) transliteration errors and (2) forms not in the transducer. We categorise three subtypes of the latter error type:

(a) correct forms — where an unanalysed form is a correct form of the language, but not in the transducer (most commonly proper nouns);
(b) spelling differences — where an unanalysed form is spelled differently in different orthographies (e.g., Kyrgyz Cyrillic ‹Троастагы› /troɑstaʁɯ/ ‘the one in Troas’ corresponds to Perso-Arabic script ‹تئرواستاعى› /tiroɑstaʁɯ/);
(c) invalid forms — where an unanalysed form is not a valid form in the language, and thus correctly does not receive an analysis (e.g., typos, misspellings, poor tokenisation, words from another language).

[6] When this paper was presented at Tu+5, Uyghur Cyrillic was measured to have naïve coverage of only 71.07%, but work since then has increased transliteration accuracy significantly. Thanks to Connor Mayer for his contributions.

Table 5: Qualitative evaluation of two transducers.

	Crimean Tatar Cyrillic	Kyrgyz Perso-Arabic
transliteration error	51	7
correct form	301	347
spelling difference	2	3
invalid form	32	3
total	386 / 1200	360 / 1200
	32.17%	30.00%

Of the four possibilities for unanalysed forms, transliteration errors are of the most concern to this paper. Correct forms may simply be added to the transducer, and their absence affects the base transducer equally, and invalid forms are true negatives whose presence simply reflects a lower level of cleanliness of the corpus. The Crimean Tatar Cyrillic analyser has a much higher proportion of transliteration errors (51 of 1200 forms) than the Kyrgyz Perso-Arabic analyser (7 of 1200 forms). Both groups of errors need further investigation in order to understand the deficiencies of the respective transcribers so that the accuracy of the transcriber transducers may be increased.

Spelling differences, while rare in these two corpora, are also not currently handled well. One solution to handle these forms correctly might be to include two different entries in the source transducer and include a comment by each one which allows the appropriate one to removed (e.g., using a tool like `grep`) before compiling each version of the transducer.

It is worth noting that the reason that the coverage on these 1200 random forms (~70%) is lower than the coverage of the transducers (>90%) is because all 1200 random forms were unique—that is, more common forms that are analysed by the transducer and have higher distribution within the corpus were only sampled once each from the corpus.

5. Conclusion. This paper presents the implementation of support for additional orthographies for seven Turkic-language transducers. Our approach (section 2) is to hand-write finite-state models of the mapping between the orthography of the base transducer and other orthographies, which is sufficient for most cases we dealt with. The transcription transducer is then intersected with the base transducer to create transducers that can analyse and generate additional orthographies. All of this work is available under Free/Open Source licenses.

Our evaluation (section 4) shows that most of the transducers implemented with transcribers have comparable naïve coverage results as the base transducer, indicating that the generalisations of the transcribers are close to complete. A look at where potential errors originate shows that a small number of incorrectly unanalysed forms are due to deficiencies of the transducers. Future work will be focused on refining the transcription transducers.

Furthermore, while errors in orthographic conversion may hypothetically yield [incorrect] analyses for forms in the additional orthography that would not receive an analysis in the base transducer's orthography, preliminary investigation of this possibility has shown that this problem is extremely rare—if not non-existent—in each of the transducers.

Several challenges that we encountered while implementing the transcription transducers were presented in section 3. Some of the potential solutions have not yet been fully implemented, and will likely increase coverage once entirely dealt with. Many new challenges, while affecting fewer forms than the ones already identified, are likely to arise while examining the [few] remaining inaccuracies in transcription transducers; identifying and solving those will constitute more future work.

One challenge identified by the evaluation was the [fairly rare, but existent] issue of different equivalent spellings in different orthographies. A related problem is different vocabulary used in different

regions, corresponding to the use of different scripts. Both of these issues should be fairly trivial to solve, but have not been dealt with yet—solutions to these problems will be included in future work.

Future work will further develop IPA transcribers for each language. Currently these are in early stages of development and face limitations for evaluation, but otherwise use the same methodology and show the same promise. We also hope to expand our methodology to more orthographies for Turkic languages, and more languages for which FOS transducers exist.

References.

Bao, Feilong, Guanglai Gao, Xueliang Yan & Hongwei Wang. 2013. Language model for Cyrillic Mongolian to Traditional Mongolian conversion. In G. Zhou, J. Li, D. Zhao & Y. Feng (eds.), *Natural Language Processing and Chinese Computing*, vol. 400 (Communications in Computer and Information Science). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-41644-6_2.

Bimba, Andrew, Norisma Idris, Norazlina Khamis & Nurul Fazmidar Mohd Noor. 2016. Stemming Hausa text: using affix-stripping rules and reference look-up. *Language Resources and Evaluation* 50. 687–703. https://doi.org/10.1007/s10579-015-9311-x.

Dovudov, Gulshan & Vít Baisa. 2010. Morphological analysis of Tajik: Notes and preliminary results. In Petr Sojka & Aleš Horák (eds.), *Proceedings of the fourth workshop on recent advances in Slavonic natural language processing (RASLAN 2010)*, 21–27.

Dovudov, Gulshan, Vít Suchomel & Pavel Šmerk. 2012. POS annotated 50M corpus of Tajik language. In Guy De Pauw, Gilles-Maurice de Schryver, Mikel L. Forcada, Kepa Sarasola, Francis M. Tyers & Peter Waiganjo Wagacha (eds.), *Proceedings of the workshop on language technology for normalisation of less-resourced languages (SALTMIL8/AfLaT2012)*, 93–98.

Eryiğit, Gülşen & Eşref Adalı. 2004. An affix stripping morphological analyzer for Turkish. In M.H. Hamza (ed.), *Proceedings of the iasted international conference on artificial intelligence and applications*, 200–304.

Forcada, Mikel L. 2001. Corpus-based stochastic finite-state predictive text entry for reduced keyboards: Application to Catalan. In *Procesamiento del lenguaje natural*, 27–65. Sociedad Española para el Procesamiento del Lenguaje Natural. http://hdl.handle.net/10045/1783.

Forcada, Mikel L., Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez & Francis M. Tyers. 2011. Apertium: A free/open-source platform for rule-based machine translation. *Machine Translation* 25. 127–144.

Johnson, Ryan, Lene Antonsen & Trond Trosterud. 2013. Using finite state transducers for making efficient reading comprehension dictionaries. In *Proceedings of the 19th Nordic conference of computational linguistics (NODALIDA 2013)*, vol. 85, 59–71.

Kann, Katharina, Ryan Cotterell & Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, 961–967. Austin, Texas: Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1097. https://www.aclweb.org/anthology/D16-1097.

Katinskaia, Anisia, Javad Nouri & Roman Yangarber. 2018. Revita: A language-learning platform at the intersection of ITS and CALL. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA). https://www.aclweb.org/anthology/L18-1644.

Kornai, András. 2013. Digital language death. *PLOS ONE* 8(10). 1–11. https://doi.org/10.1371/journal.pone.0077056.

Linden, Krister, Miikka Silfverberg, Erik Axelson, Sam Hardwick & Tommi Pirinen. 2011. HFST—Framework for Compiling and Applying Morphologies. In Cerstin Mahlow & Michael Pietrowski

(eds.), *Systems and Frameworks for Computational Morphology*, vol. 100 (Communications in Computer and Information Science), 67–85.

Lindén, Krister, Erik Axelson, Senka Drobac, Sam Hardwick, Miikka Silfverberg & Tommi A. Pirinen. 2013. Using HFST for creating computational linguistic applications. In A. Przepiórkowski, M. Piasecki, K. Jassem & P. Fuglewicz (eds.), *Computational linguistics: Applications*, vol. 458 (Studies in Computational Intelligence), 3–25. Berlin, Heidelberg: Springer.

Liu, Zhinan, Feilong Bao, Guanglai Gao & Suburi. 2018. Mongolian grapheme to phoneme conversion by using hybrid approach. In M. Zhang, V. Ng, D. Zhao, S. Li & H. Zan (eds.), *Natural Language Processing and Chinese Computing*, vol. 11108 (Lecture Notes in Computer Science). Cham: Springer. https://doi.org/10.1007/978-3-319-99495-6_4.

Mayer, Connor, Travis Major & Mahire Yakup. N.d. Conflicting trigger effects in Uyghur backness harmony. To appear.

Pirinen, Tommi A. & Krister Lindén. 2014. State-of-the-art in weighted finite-state spell-checking. In Alexander Gelbukh (ed.), *Computational linguistics and intelligent text processing: Proceedings of the 15th international conference, cicling 2014*, vol. 8404.II (Lecture Notes in Computer Science), 519–532. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-54903-8_43.

Saini, Tejinder Singh & Gurpreet Singh Lehal. 2008. Shahmukhi to Gurmukhi transliteration system: A corpus based approach. *Advances in Natural Language Processing and Applications* 33. 151–162.

Silfverberg, Miikka, Mirka Hyvärinen & Tommi Pirinen. 2011. Improving predictive entry of Finnish text messages using IRC logs. In K. Jassem, P. Fuglewicz, M. Piasecki & A. Przepiórkowski (eds.), *Proceedings of the computational linguistics-applications conference*, 69–76.

Silfverberg, Miikka & Francis Tyers. 2019. Data-driven morphological analysis for Uralic languages. In *Proceedings of the fifth international workshop on computational linguistics for Uralic languages*, 1–14. Tartu, Estonia: Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-0301. https://www.aclweb.org/anthology/W19-0301.

Tyers, Francis M., Tommi Pirinen & Jonathan N. Washington. 2015. Finite-state morphologies and text corpora as resources for improving morphological descriptions. In *Workshop on Computational Phonology and Morphology at the Linguistic Summer Institute*. Linguistic Society of America.

Tyers, Francis M., Jonathan Washington, Darya Kavitskaya, Memduh Gökırmak, Nick Howell & Remziye Berberova. 2019. A biscriptual morphological transducer for Crimean Tatar. In vol. 1, 74–80. https://scholar.colorado.edu/scil-cmel/vol1/iss1/10.

Washington, Jonathan N., Aziyana Bayyr-ool, Aelita Salchak & Francis M. Tyers. 2016. Development of a finite-state model for morphological processing of Tuvan. *Родной Язык* 1(4). 156–187. http://rodyaz.ru/pdf/no.4_2016/Washington%20J.,%20Bayyr-ool%20A.,%20Salchak%20A.,%20Tyers%20F.%20Development%20of%20a%20finite-state%20model%20for%20morphological%20processing%20of%20Tuvan.pdf.

Washington, Jonathan N., Ilnar Salimzianov, Francis M. Tyers, Memduh Gökırmak, Sardana Ivanova & Oğuzhan Kuyrukçu. 2020. Free/open-source technologies for Turkic languages developed in the Apertium project. In *Proceedings of the seventh international conference on Turkic language processing (TurkLang 2019)*. in press.