Out-of-the-box Universal Romanization Tool \textit{uroman}

Ulf Hermjakob, Jonathan May, Kevin Knight
Information Sciences Institute, University of Southern California
\{ulf,jonmay,knight\}@isi.edu

Abstract

We present \textit{uroman}, a tool for converting text in myriads of languages and scripts such as Chinese, Arabic and Cyrillic into a common Latin-script representation. The tool relies on Unicode data and other tables, and handles nearly all character sets, including some that are quite obscure such as Tibetan and Tifinagh. \textit{uroman} converts digital numbers in various scripts to Western Arabic numerals. Romanization enables the application of string-similarity metrics to texts from different scripts without the need and complexity of an intermediate phonetic representation. The tool is freely and publicly available as a Perl script suitable for inclusion in data processing pipelines and as an interactive demo web page.

1 Introduction

String similarity is a useful feature in many natural language processing tasks. In machine translation, it can be used to improve the alignment of bitexts, and for low-resource languages with a related language of larger resources, it can help to decode out-of-vocabulary words. For example, suppose we have to translate \textit{degustazione del vino} without any occurrence of \textit{degustazione} in any training corpora, but we do know that in a related language \textit{dégustation de vin} means \textit{wine tasting}, we can use string similarity to infer the meaning of \textit{degustazione}.

String similarity metrics typically assume that the strings are in the same script, but many cross-lingual tasks such as machine translation often involve multiple scripts. If we can romanize text from a non-Latin script to Latin, standard string similarity metrics can be applied, including edit distance-based metrics (Levenshtein, 1966; Winkler, 1990) and phonetic-based metrics such as Metaphone (Philips, 2000).

Hindi, for example, is written in the Devanagari script and Urdu in the Arabic script, so any words between those two languages will superficially appear to be very different, even though the two languages are closely related. After romanization, however, the similarities become apparent, as can be seen in Table 1:

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
Original & Hindi & Urdu & English \\
\hline
\text{नेपाल} & \text{Nepal} & \text{nepal} & \text{Nepal} \\
\hline
\end{tabular}
\caption{Example of Hindi and Urdu romanization}
\end{table}

Foreign scripts also present a massive cognitive barrier to humans who are not familiar with them. We devised a utility that allows people to translate text from languages they don’t know, using the same information available to a machine translation system (Hermjakob et al., 2018). We found that when we asked native English speakers to use this utility to translate text from languages such as Uyghur or Bengali to English, they strongly preferred working on the romanized version of the source language compared to its original form and indeed found using the native, unfamiliar script to be a nearly impossible task.

1.1 Scope of Romanization

Romanization maps characters or groups of characters in one script to a character or group of characters in the Latin script (ASCII) with the goal to approximate the pronunciation of the original text and to map cognates in various languages to similar words in the Latin script, typically without the
use of any large-scale lexical resources. As a secondary goal, romanization standards tend to prefer reversible mappings. For example, as standalone vowels, the Greek letters ι (iota) and υ (upsilon) are romanized to i and y respectively, even though they have the same pronunciation in Modern Greek.

uroman generally follows such preference, but uroman is not always fully reversible. For example, since uroman maps letters to ASCII characters, the romanized text does not contain any diacritics, so the French word ou (“or”) and its homophone o`u (“where”) both map to romanized ou.

uroman provides the option to map to a plain string or to a lattice of romanized text, which allows the system to output alternative romanizations. This is particularly useful for source languages that use the same character for significantly different sounds. The Hebrew letter Pe for example can stand for both p and f. Lattices are output in JSON format.

Note that romanization does not necessarily capture the exact pronunciation, which varies across time and space (due to language change and dialects) and can be subject to a number of processes of phonetic assimilation. It also is not a translation of names and cognates to English (or any other target language). See Table 3 for examples for Greek.

A romanizer is not a full transliterator. For example, this tool does not vowelize text that lacks explicit vowelization such as normally occurring text in Arabic and Hebrew (i.e., without diacritics/points); see Table 4.

### Table 2: Romanization examples for 10 scripts

<table>
<thead>
<tr>
<th>Language</th>
<th>Original</th>
<th>Romanization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amharic</td>
<td>ኢማርሱ ራለታ ብታት  ከተለ ዝወል።</td>
<td>እማርሱ ይላን ለን ዝወል።</td>
</tr>
<tr>
<td>Arabic</td>
<td>المملكة العربية السعودية</td>
<td>الامپرية العربية السعودية</td>
</tr>
<tr>
<td>Greek</td>
<td>Γερουτάν Ντάσιλμπουμ</td>
<td>Γερουτάν Ντάσιλμπουμ</td>
</tr>
<tr>
<td>Hebrew</td>
<td>נְחֶבֶת בירוטליא</td>
<td>נְחֶבֶת בירוטליא</td>
</tr>
<tr>
<td>Japanese</td>
<td>アメリカ</td>
<td>アメリカ</td>
</tr>
<tr>
<td>Korean</td>
<td>세계에서 6번째로 연혁이 난은 나라이다.</td>
<td>세계에서 6번째로 연혁이 난은 나라이다.</td>
</tr>
<tr>
<td>Mandarin</td>
<td>北卡罗来纳</td>
<td>北卡罗来纳</td>
</tr>
<tr>
<td>Nepali</td>
<td>तिब्बती पाण्डवा वास्त्रो नाम</td>
<td>तिब्बती पाण्डवा वास्त्रो नाम</td>
</tr>
<tr>
<td>Tamil</td>
<td>கோட்டு கரமநாகராக்கள்</td>
<td>கோட்டு கரமநாகராக்கள்</td>
</tr>
<tr>
<td>Tibetan</td>
<td>བོད་ཐང་པོ་</td>
<td>བོད་ཐང་པོ་</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Language</th>
<th>Original</th>
<th>Romanization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modern Greek</td>
<td>Κρήτη γεωλογία μπανάνα</td>
<td>Κρήτη γεωλογία μπανάνα</td>
</tr>
<tr>
<td>Pronunciation</td>
<td>Kriti yeologyia banana</td>
<td>Kriti yeologyia banana</td>
</tr>
<tr>
<td>Romanization</td>
<td>Kreta geologia banana</td>
<td>Kreta geologia banana</td>
</tr>
<tr>
<td>English</td>
<td>Crete geology banana</td>
<td>Crete geology banana</td>
</tr>
<tr>
<td>German</td>
<td>Kreta Geologie Banane</td>
<td>Kreta Geologie Banane</td>
</tr>
</tbody>
</table>

### Table 3: Examples of Greek romanization

| Table 4: Romanization with and without diacritics |
|------------------|------------------|------------------|
| Arabic           | Normal text      | With diacritics  |
| Romanization     | washntn          | wasshintum       |
| English          | Washington       | Washington       |

### 1.2 Features

uroman has the following features:

1. Input: UTF8-encoded text and an optional ISO-639-3 language code
2. Output: Romanized text (default) or lattice of romanization alternatives in JSON format
3. Nearly universal romanization
4. N-to-m mapping for groups of characters that are non-decomposable with respect to romanization
5. Context-sensitive and source language-specific romanization rules

1See Section 4 for a few limitations.
Romanization tools have long existed for specific individual languages such as the Kakasi\(^2\) kanji-to-kana/romaji converter for Japanese, but to the best of our knowledge, we present the first publicly available (near) universal romanizer that handles n-to-m character mappings. Many romanization examples are shown in Table 2 and examples of n-to-m character mapping rules are shown in Table 7.

### 2 System Description

#### 2.1 Unicode Data

As its basis, *uroman* uses the character descriptions of the Unicode table.\(^3\) For the characters of most scripts, the Unicode table contains descriptions such as *CYRILLIC SMALL LETTER SHORT U* or *CYRILLIC CAPITAL LETTER TE WITH MIDDLE HOOK*. Using a few heuristics, *uroman* identifies the phonetic token in that description, i.e. *U* and *TE* for the examples above. The heuristics use a list of anchor keywords such as *letter* and *syllable* as well as a number of modifier patterns that can be discarded. Given the phonetic token of the Unicode description, *uroman* then uses a second set of heuristics to predict the romanization for these phonetic tokens, i.e. *u* and *t*. For example, if the phonetic token is one of more consonants followed by one or more vowels, the predicted romanization is the leading sequence of consonants, e.g. SHA → *sh*.

#### 2.2 Additional Tables

However, these heuristics often fail. An example of a particularly spectacular failure is SCHWA → *schw* instead of the desired *e*. Additionally, there are sequences of characters with non-compositional romanization. For example, the standard romanization for the Greek sequence omikron+upsilon, *(οι)* is the Latin *ou* rather than the character-by-character romanization *oy*.

As a remedy, we manually created additional correction tables that map sequences of one or more characters to the desired romanization, with currently 1,088 entries. The entries in these tables can be restricted by conditions, for example to specific languages or to the beginning of a word, and can express alternative romanizations. This data table is a core contribution of the tool.

*uroman* additionally includes a few special heuristics cast in code, such as for the vowelizations of a number of Indian languages and Tibetan, dealing with diacritics, and a few language-specific idiosyncrasies such as the Japanese sokuon and Thai consonant-vowel swaps.

Building these *uroman* resources has been greatly facilitated by information drawn from Wikipedia,\(^4\) Richard Ishida’s script notes,\(^5\) and ALA-LC Romanization Tables.\(^6\)

#### 2.3 Characters without Unicode Description

The Unicode table does not include character descriptions for all scripts.

For Chinese characters, we use a Mandarin pinyin table for romanization.

For Korean, we use a short standard Hangul romanization algorithm.\(^7\)

For Egyptian hieroglyphs, we added single-sound phonetic characters and numbers to uroman’s additional tables.

#### 2.4 Numbers

*uroman* also romanizes numbers in digital form.

For some scripts, number characters map one-to-one to Western Arabic numerals 0-9, e.g. for Bengali, Eastern Arabic and Hindi.

For other scripts, such as Amharic, Chinese, and Egyptian hieroglyphics, written numbers are structurally different, e.g. the Amharic number character sequence 10·9·100·90·8 = 1998 and the Chinese number character sequence 2·10·5·10000·6·1000 = 256000. *uroman* includes a special number module to accomplish this latter type of mapping. Examples are shown in Table 5.

Note that for phonetically spelled-out numbers such as Greek οκτώ, *uroman* romanizes to the spelled-out Latin *okto* rather than the digital 8.

\(^{2}\)http://kakasi.namazu.org

\(^{3}\)ftp://ftp.unicode.org/Public/UNIDATA/UnicodeData.txt

\(^{4}\)https://en.wikipedia.org

\(^{5}\)https://r12a.github.io/scripts/featurelist

\(^{6}\)https://www.loc.gov/catdir/cpso/roman.html

\(^{7}\)http://gernot-katzers-spice-pages.com/var/korean_hangul_unicode.html
2.5 Romanization of Latin Text

Some Latin script-based languages have words for which spelling and pronunciation differ substantially, e.g. the English name *Knight* (IPA: /kaɪt/) and French *Bordeaux* (boʁ. do), which complicates string similarity matching if the corresponding spelling of the word in the non-Latin script is based on pronunciation.

*uroman* therefore offers alternative romanizations for words such as *Knight* and *Bordeaux* (see Table 6 for an example of the former), but, as a policy *uroman* always preserves the original Latin spelling, minus any diacritics, as the top romanization alternative.

Table 6: Romanization with alternatives

<table>
<thead>
<tr>
<th>Original</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>矢田</td>
<td>矢田</td>
</tr>
<tr>
<td>Top romanization</td>
<td>nait</td>
</tr>
<tr>
<td>An alternative romanization</td>
<td>nait</td>
</tr>
<tr>
<td>Alternative lattice</td>
<td></td>
</tr>
</tbody>
</table>

Table 7 includes examples of the Romanization rules in *uroman*, including n-to-m mappings.

2.6 Caching

*uroman* caches token romanizations for speed.
dom texts. After picking the first random text, additional random texts will be available from three corpora to choose from (small, large, and Wikipedia articles about the US). Users can then restrict the randomness in a special option field. For example, \texttt{--l} will exclude texts in Latin+ scripts. For further information about possible restrictions, hover over the word restriction (a dotted underline indicates that additional info will be shown when a user hovers over it).

The romanization of the output at the demo site is mouse sensitive. Hovering over characters of either the original or romanized text, the page will highlight corresponding characters. See Figure 1 for an example. Hovering over the original text will also display additional information such as the Unicode name and any numeric value. To support this interactive demo site, the \textit{uroman} package also includes fonts for Burmese, Tifinagh, Klin- gson, and Egyptian hieroglyphs, as they are sometimes missing from standard browser font packages.

4 Limitations and Future Work
The current version of \textit{uroman} has a few limitations, some of which we plan to address in future versions. For Japanese, \textit{uroman} currently romanizes hiragana and katakana as expected, but kanji are interpreted as Chinese characters and romanized as such. For Egyptian hieroglyphs, only single-sound phonetic characters and numbers are currently romanized. For Linear B, only phonetic syllabic characters are romanized. For some other extinct scripts such as cuneiform, no romanization is provided.

\textit{uroman} allows the user to specify an ISO-639-3 source language code, e.g. \texttt{uig} for Uyghur. This invokes any language-specific romanization rules for languages that share a script with other languages. Without source language code specification, \textit{uroman} assumes a default language, e.g. Arabic for text in Arabic script. We are considering adding a source language detection component that will automatically determine whether an Arabic-script source text is Arabic, Farsi, or Uyghur etc.

5 Romanization Applications
5.1 Related Work
Gey (2009) reports that romanization based on ALA-LC romanization tables (see Section 2.2) is useful in cross-lingual information retrieval.

There is a body of work mapping text to phonetic representations. Deri and Knight (2016) use Wiktionary and Wikipedia resources to learn text-to-phoneme mappings. Phonetic representations are used in a number of end-to-end transliteration systems (Knight and Graehl, 1998; Yoon et al., 2007). Qian et al. (2010) describe the toolkit ScriptTranscriber, which extracts cross-lingual transliteration pairs from comparable corpora. A core component of ScriptTranscriber maps text to an ASCII variant of the International Phonetic Alphabet (IPA).

Andy Hu’s transliterator is a fairly universal romanizer in JavaScript, limited to romanizing one Unicode character at a time, without context.

5.2 Applications Using \textit{uroman}

\textit{uroman} has also been used in our aforementioned translation utility (Hermjakob et al., 2018), where humans translate text to another language, with computer support, with high fluency in the target language (English), but no prior knowledge of the source language.

\textit{uroman} has been partially ported by third parties to Python and Java.\footnote{https://github.com/andyhu/transliteration}

6 Conclusion
Romanization tools have long existed for specific individual languages, but to the best of our knowledge, we present the first publicly available (near) universal romanizer that handles n-to-m character mappings. The tool offers both simple plain text as well as lattice output with alternatives, and includes romanization of numbers in digital form. It has been successfully deployed in a number of multi-lingual natural language systems.

Acknowledgment
This work is supported by DARPA (HR0011-15-C-0115).

\footnote{https://github.com/andyhu/transliteration}

\footnote{https://github.com/BBN-E/bbn-transliterator}
References


