Epitran: Precision G2P for Many Languages

David R. Mortensen, Siddharth Dalmia, Patrick Littell
Language Technologies Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
dmortens@cs.cmu.edu, sdalmia@andrew.cmu.edu, plittell@cs.cmu.edu

Abstract

Epitran is a massively multilingual, multiple back-end system for G2P (grapheme-to-phoneme) transduction which is distributed with support for 61 languages. It takes word tokens in the orthography of a language and outputs a phonemic representation in either IPA or X-SAMPA. The main system is written in Python and is publicly available as open source software. Its efficacy has been demonstrated in multiple research projects relating to language transfer, polyglot models, and speech. In a particular ASR task, Epitran was shown to improve the word error rate over Babel baselines for acoustic modeling.

Keywords: G2P, grapheme, phoneme, International Phonetic Alphabet, IPA, X-SAMPA, multilingual, speech, transfer, polyglot

1. Introduction

Epitran is a massively multilingual G2P system. To maximize its usefulness, it is written in Python and distributed as open source software under an MIT license. Out of the box, it supports 61 languages. Additional languages can easily be added using either a simple, rule-based framework or by adding other back-ends. It has a number of advantages over other G2P and romanization packages like Unitran and URoman including sensible handling of different Latin scripts, precision transduction for each language-script pair (important when multiple languages use the same script differently), and proper use of international and *de facto* standards for phonetic representation (IPA, X-SAMPA).

2. Motivation and Related Work

A number of Speech and NLP tasks require a G2P step—the conversion of orthographic representations to phonemic or phonetic representations (Daelemans and van den Bosch, 1996; Black et al., 1998; Sproat, 2006; Bisani and Ney, 2008; Laurent et al., 2009; Rao et al., 2015). Such tasks include TTS, ASR, the training of polyglot phonetic models for (non-speech) NLP tasks, and the implementation of approximate phonetic matching. G2P research has focused on some of the most difficult cases, including English. Less attention has been paid to providing G2P coverage for the wide range of languages with a more direct relationship between orthography and phonology. Many such languages require more than simple a mapping table for G2P but can be handled adequately by a more sophisticated rule-based system and certainly do not require a machine learning approach. This is fortunate, because many of these are also low resource languages in which sufficient training data, even for training a WFST, are not available but adequate linguistic descriptions do exist.

There are a few tools that already occupy this niche. Unitran is a tool that converts orthographic text to WorldBet and X-SAMPA (Qian et al., 2010). It is limited, though, in that it does not support Roman scripts and it does not have a mechanism for specifying different behavior for different languages that use the same script. For example, Amharic and Tigrinya both use the Ethiopic script and are treated exactly the same by Unitran even though their use of the Ethiopic script differs somewhat. URoman[1] is not a true G2P system; rather, it is a romanizer which generates an English-compatible romanization of the orthographic input rather than a phonemically sound output using an explicitly formalized standard (like WorldBet, X-SAMPA, or IPA). This makes it very useful for translating data into a form that English-speaking researchers can easily read and makes it somewhat useful for phonetically-driven entity linking tasks but limits its usefulness in speech and other tasks where the phonetic identity of segments is important. While Epitran is more limited in its coverage than Unitran and particularly URoman, it provides precision G2P that is easy to augment and debug on a language-by-language basis. It outputs both IPA—which can be consumed by related tools like PanPhon (Mortensen et al., 2016)—and X-SAMPA, which is now widely used in crosslingual speech applications.

A well-constructed Epitran mode will return a plausible IPA output for every well-formed input token. This provides a clear advantage over lexically-based resources for grapheme-to-phoneme tasks. This means there are effectively no vocabulary limits—there is a pronunciation for every word presented. The advantages are even more clear when compared to manually annotated data, as is often used in speech tasks. It is not uncommon for multiple annotators to have a low degree of agreement, particularly for phonemes that are phonetically close (differing in few articulatory features). This can cause inconsistency in training labels leading to a bad ASR engine. Epitran addresses this problem by providing a perfectly consistent mapping between graphemes and phonemes.

Epitran also places all converted data in a common phonetic space. This is useful when working with data coming from different sources (including data written in different scripts), or different languages and makes certain types of polyglot models possible.

It is reasonable to ask whether G2P is necessary any longer. After all, it has been shown that a respectable ASR system can be constructed for Vietnamese working directly from the orthography (Luong and Vu, 2016). Two points must be made here: Epitran is focused primarily at languages in

[1] `https://www.isi.edu/~ulf/uroman.html`

low data scenarios (though high resource languages are included for transfer purposes). With very little data, the G2P step becomes important. Relationships between orthography and pronunciation, which might be "obvious" to a system in a high data setting, are less obvious when data resources are constrained. Furthermore, the use of G2P is not limited to speech tasks alone. There are other NLP tasks, like approximate phonetic matching, where grapheme-to-phoneme conversion are important. Perhaps most significantly, certain models of cross-lingual transfer are facilitated by placing data from related languages in a common phonetic space (Bharadwaj et al., 2016). This topic is taken up at greater length in §6.2.

3. Interface

Epitran provides a simple Python (2 and 3) interface to all backends (See §4.):

```
>>> import epitran
>>> epi = epitran.Epitran('eng-Latn')
>>> epi.transliterate('Berkeley')
'bɹ̩kli'
>>> epi.xsampa_list('Berkeley')
['b','r\\=','k','l','i']
```

The primary API consists of methods on `Epitran` objects. The most important of these are `transliterate` and `xsampa_list`. The `transliterate` method takes a word token and returns its IPA representation. The `xsampa_list` method takes a word token a returns a list of X-SAMPA segments. Other methods include `trans_list` (like `transliterate` but returns a list of IPA segments) and `trans_delimiter` (like `transliterate` but returns IPA with a delimiter between segements).

4. Architecture

Epitran is structured as a single Python interface to various backends. The current backends are:

- **Flite** (for English)
- **Epihan** (for Simplified and Traditional Chinese)
- **Simple Epitran** (for all other languages)

The Flite backend requires the installation of the `lex_lookup` binary from a recent version of the Flite Speech Synthesis System (Black and Lenzo, 2001). The Epihan backend requires the installation of the open source CC-CEDict dictionary[2]. Most of the languages supported require the installation of no software or data other than Epitran itself. These use the Simple Epitran backend.

4.1. Simple Epitran

Every language with Simple Epitran support has a map file that defines a mapping between orthographic strings and phonemic strings. This is easy to produce and maintain. In many cases, it can be extracted automatically or semi-automatically from tables in existing resources. When it is not adequate, a preprocessor and a postprocessor can be added to manipulate the representations before and after mapping.

[2] `https://cc-cedict.org`

```
::consonant:: = b|bw|d|dw|d͡ʒ|d͡ʒw|f|fʲ|fw|h|hw|j|k|kʷ
|l|lw|m|mʲ|mw|n|nw|p|pw|pʼ|pʼw|q|qʰ|qʰw|qʷ|r|rʲ|rw|s|sw
|t|tw|tʼ|tʼw|t͡sʼ|t͡sʼw|t͡ʃ|t͡ʃw|t͡ʃʼ|t͡ʃʼw|v|vw|w|x|xʷ
|z|zw|ħ|ħw|ŋ|ŋw|g|gʷ|ɲ|ɲw|ʃ|ʃw|ʒ|ʒw|ʔ|ʔw|ʕ

0 -> ɨ / #(::consonant::)_(::consonant::)
0 -> ɨ / #(::consonant::)_#
0 -> ɨ / (::consonant::)_(::consonant::)#
0 -> ɨ / (::consonant::)_(::consonant::)(::consonant::)
```

Figure 1: The postprocessor from a Tigrinya (`tir-Ethi-red`) mode.

4.1.1. Map Files

Simple Epitran map files are simply two-column CSV files with a field for orthographic representation and a field for phonemic representation. They are interpreted greedily: The longest orthographic string matching a prefix of the input string is removed from the input string and the corresponding phonemic string is appended to the output string. This process proceeds iteratively until the input string is consumed. By default, input characters that are not found in the mapping table are added to the output unchanged. The order of the pairs in the map file is not significant.

4.1.2. Pre- and Post-Processors

While map files are easy to produce, often with the kinds of data available in references like Wikipedia[3] and Omniglot [4] as well as published grammars, they are not adequate for languages with complex mappings between orthography and phonology. For these circumstances, Epitran provides preprocessors and postprocessors. Each of these is essentially a cascade of context-sensitive rewrite rules. Figure 1 illustrates the structure of Simple Epitran preprocessors and postprocessors through an example of a Tigryina postprocessor. The first line (broken into four lines here) defines a constant, `::consonant::`, that can be used in the rules. Its value is a regular expression that matches any of the consonant phonemes in the language. The rules insert the default vowel /ɨ/ between two word-initial consonants, at the end of a word consisting only of a consonants, between word-final consonants, and between the second and third consonant of a three-consonant sequence. The function of this postprocessor in the Simple Epitran pipeline is illustrated in Figure 2. The input orthographic string passes through the preprocessor unchanged. It is then remapped into a phonemic space as defined in the mapping table. However, this representation does not include certain vowels whose distribution is predictable. One such vowel is inserted into 'bee', yielding /nɨhbi/ as the output.

4.2. Design Decisions

It should be evident that the preprocessors, postprocessrs, and maps all define regular relations and can thus be modeled using Finite State Transducers. Indeed, at an early stage of development, we considered using XFST or Foma as a basis for Epitran. There were three reasons we ulti-

[3] `http://www.wikipedia.org`
[4] `https://www.omniglot.com`

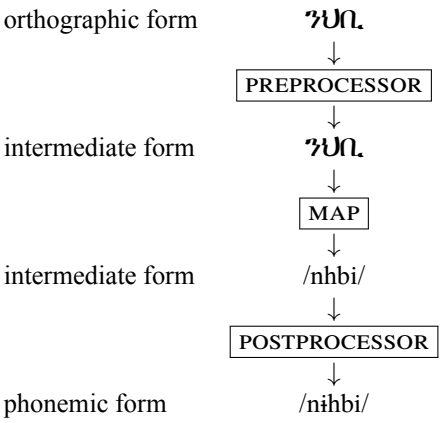

Figure 2: The Simple Epitran pipeline illustrated by the Tigrinya word ንህቢ ‘bee’.

mately chose not to do this (and to adopt—for Simple Epitran, at least—a pure Python implementation using string operations and regular expressions):

1. We wished to minimize the set of dependencies for Simple Epitran and wanted to limit them to Python libraries that could be installed from PyPI with `pip`.

2. We wanted a specific, lightweight format for the map files that could easily be extracted from sources with minimal processing. The greedy matching described above could be achieved through composing finite state transducers such that “rules” with a longer left-hand side would be ordered first, but this would require programmatic reordering of the list of mappings.

3. Some languages require metathesis (the “swapping” of substrings) in order to map orthographic order onto the temporal order needed for phonemic representation. It is possible to implement this using FSTs, but it is awkward to express using XFST/Foma’s rule notation. In Python, we were able to implement syntactic sugar that makes general metathesis rules easy to write.

5. Current Coverage

Epitran currently has coverage over the languages in Table 1. Languages annotated as “Provisional” need additional testing. “Naive phonemic” modes naively assume that the orthography of the language is transparently phonemic. A few languages written with Perso-Arabic scripts (Arabic, Persian, Urdu) are rendered without short vowels (which are not present in the orthographies). Reduced inventory modes are alternate modes for a language-script pair that reanalyze the phonemic system to minimize the number of phoneme types in the output.

Variation is an issue that inevitably arises in any attempt to model pronunciation. For Spanish, for example, their are numerous varieties that realize the same orthographic consonants differently. European varieties differ among themselves and also differ from Latin American varieties. For example ⟨z⟩ is realized as [θ] in Northern and Central Spain, but as [s] elsewhere in the Spanish-speaking world. Similarly, ⟨ll⟩ may be realized phonetically as [ʎ], [j̟] or [d͡ʒ], depending on the region. The mode `spa-Latn` represents a compromise Latin American Spanish, for historical reasons. In principle, there is no reason that a Castilian Spanish mode could not be added with modifying suffix appended to the language-script code. In practice, one variety—or a compromise variety—has been chosen for each ISO 639-3 code. In some cases, the choice of a compromise variety has been influenced by certain use cases. For example, the

Code	Language (Script)	Notes
aar-Latn	Afar	
amh-Ethi	Amharic	Naive mode
amh-Ethi-pp	Amharic	Precise phonemic mode
amh-Ethi-red	Amharic	Reduced inventory mode
ara-Arab	Arabic	No short vowels
aze-Cyrl	Azerbaijani (Cyrillic)	
aze-Latn	Azerbaijani (Latin)	
ben-Beng	Bengali	
cat-Latn	Catalan	
ceb-Latn	Cebuano	
cmn-Hans	Chinese (Simplified)	
cmn-Hant	Chinese (Traditional)	
ckb-Arab	Sorani	Provisional
deu-Latn	German	
deu-Latn-np	German	Naive phonemic
eng-Latn	English	
fas-Arab	Farsi	No short vowels
fra-Latn	French	Provisional
fra-Latn-np	French	Naive phonemic
hat-Latn-bab	Haitian	Following Babel
hau-Latn	Hausa	
hin-Deva	Hindi	
hun-Latn	Hungarian	
ilo-Latn	Ilocano	
ind-Latn	Indonesian	
ita-Latn	Italian	
jav-Latn	Javanese	
kaz-Cyrl-bab	Kazakh (Cyrillic)	Following Babel
kaz-Cyrl	Kazakh (Cyrillic)	
kaz-Latn	Kazakh (Latin)	
khm-Khmr	Khmer	Provisional
kin-Latn	Kinyarwanda	
kir-Arab	Kyrgyz (Perso-Arabic)	
kir-Cyrl	Kyrgyz (Cyrillic)	
kir-Latn	Kyrgyz (Latin)	
kmr-Latn	Kurmanji	
lao-Laoo	Lao	Provisional
mar-Deva	Marathi	
mon-Cyrl-bab	Mongolian	Following Babel
mlt-Latn	Maltese	
msa-Latn	Malay	
mya-Mymr	Burmese	Provisional
nld-Latn	Dutch	
nya-Latn	Chichewa	
orm-Latn	Oromo	
pan-Guru	Punjabi	
pol-Latn	Polish	
por-Latn	Portuguese	
rus-Cyrl	Russian	Provisional
sna-Latn	Shona	
som-Latn	Somali	
spa-Latn	Spanish	
swa-Latn	Swahili	
swe-Latn	Swedish	
tam-Taml	Tamil	
tel-Telu	Telugu	
tgk-Cyrl	Tajik	
tgl-Latn	Tagalog	
tha-Thai	Thai	
tir-Ethi	Tigrinya	Naive
tir-Ethi-pp	Tigrinya	Precise phonemic mode
tir-Ethi-red	Tigrinya	Reduced inventory mode
tuk-Cyrl	Turkmen (Cyrillic)	
tuk-Latn	Turkmen (Latin)	
tur-Latn-bab	Turkish	Following Babel
tur-Latn	Turkish	
uig-Arab	Uyghur	
ukr-Cyrl	Ukrainian	Provisional
urd-Arab	Urdu	No short vowels
uzb-Cyrl	Uzbek (Cyrillic)	
uzb-Latn	Uzbek (Latin)	
vie-Latn	Vietnamese	
xho-Latn	Xhosa	
yor-Latn	Yoruba	
zha-Latn	Zhuang	
zul-Latn	Zulu	

Table 1: Language modes in Epitran

⟨r⟩ phoneme in German and French has been mapped to [r] (found in conservative non-standard dialects) rather than [ʁ] or [ʀ]. This decision was made in order to minimize the phonetic distance between French and German words and related vocabulary items in other languages. Enriching the documentation to clarify the nature of these decisions is an important next step in improving Epitran's usability.

6. Downstream Evaluation and Applications

Evaluation of a multilingual resource like Epitran is difficult because each of the languages supported must effectively be evaluated separately. Furthermore, since Epitran concentrates on low resource languages, it is often difficult to find a sensible baseline and ground truth for many of the languages supported. Unitran could be used as a baseline for languages with non-Latin scripts but not for those with Latin scripts (the majority of Epitran languages). With these caveats in mind, we find good evidence from task-based evaluation that at least some of Epitran's language modes are very valuable.

6.1. ASR

To measure the performance of Epitran G2P in an ASR task, we trained an acoustic model of the EESEN ASR system (Miao et al., 2015) with a 6 layer bidirectional LSTM having 140 cells followed by a linear projection to 70 layer network after each LSTM[5]. The model takes in filter bank and pitch features extracted from telephonic audio recorded at 8kHz and produces a sequence of phonemes as the output. For the baseline experiment we used the G2P dictionary (lexicon file) provided as part of the dataset for each of the languages. For Epitran, we trained the same model by generating a new pronunciation file for all words in the baseline lexicon. For both the models we kept 5% of the data as a held out cross validation data to choose the best epoch.

We ground these phonemes to words using Weighted Finite State Transducers (WFSTs). We generate separate WFSTs to encode information about the tokens, grammar, and lexicon, and fuse them into a single compressed search graph which is used to generate the words. The details of the process can be seen in (Miao et al., 2015). This way we can compare the results on "gold transcription" text. We experimented on the eleven languages that were both in data released as part of the IARPA BABEL Research Program (IARPA-BAA-11-02) and for which there was a fully-developed Epitran mode. The word error rate results are shown in Table 2. In all cases, Epitran is competitive with the baseline. In a majority of cases, the word error rate for the system trained using Epitran has a lower word error rate that the baseline system.

One persistent advantage of Epitran (over lexical resources) is its ability to produce consistent and accurate pronunciations for all words that a system encounters, not just thus listed in the lexicon. This aids in several ASR-related tasks. For example, it makes it possible to build phoneme-based language models with no out-of-vocabulary (OOV) words. This has applications beyond speech technologies (in cross-lingual language applications). It also enables us to recognize words outside of the lexicon on which a model is trained. It can produce precise pronunciations of an utterance which can provide useful feedback on how the model is performing on OOV words during evaluation.

[5]The code to train can be found in `https://github.com/srvk/eesen/tree/tf_clean`

Language	Baseline WER	Epitran WER	Vocabulary Size
Amharic	58.6	**57.2**	36971
Cebuano	60.3	**57.1**	15534
Javanese	71.3	**65.7**	15541
Kazakh	60.7	**57.8**	22371
Kurmanji	70.9	**68.4**	14425
Swahili	**60.7**	61.2	18796
Tagalog	**54.6**	55.7	22627
Tamil	**74.8**	76.8	58484
Telugu	81.5	**77.9**	37654
Turkish	**55.7**	56.9	41157
Zulu	67.7	**65.2**	60627

Table 2: Word Error Rate (% WER) for the baseline model and a model trained with Epitran

6.2. Additional Applications

Epitran has been further applied to a number of tasks undertaken by the ARIEL team as part of the DARPA LORELEI program. In the speech domain, it has also been used in the cross-lingual and cross-domain transfer of ASR models built in Amharic to Tigrinya and Oromo. Since it performs G2P using similar rules for all the languages, the phonetic spaces to which the languages are mapped end up being very close. This is extremely important for transfer learning, especially when the target language data is minimal. It was also used to combine Amharic datasets from different sources by generating a new G2P dictionary. This technique is extremely useful for low resource languages where collecting data from multiple sources is essential. It has led to significant improvements in the robustness and performance of ARIEL's ASR.

In the language domain, it has been used to facilitate cross-lingual transfer of named entity annotations from Uzbek to Turkish and from Uzbek and Turkish to Uyghur by allowing the projection of all these languages into a common, phonetic space (Bharadwaj et al., 2016). It has also been used to transduce all languages in a large multilingual parallel corpus into IPA for training a polyglot machine translation model. Furthermore, it has been used in cross-lingual entity linking between languages like Amharic, Tigrinya, and Oromo, on the source side, and English, on the target side. Unfortunately, none of these cases were well adapted to ablation experiments in which the contribution of Epitran could be straightforwardly evaluated.

An additional application space for Epitran—and one that is even harder to evaluate—lies in helping non-native speaker linguists perform annotation tasks. While foreign scripts may be opaque even to linguistically-trained annotators, IPA representations are widely recognizable and read-

able. Epitran has been pivotal in allowing members of the ARIEL team to perform named entity annotation on languages like Uyghur, Amharic, and Tigrinya without special competence in these languages.

7. Conclusion

Epitran provides a lightweight and precise means of mapping orthographic data into the phonetic space. The out-of-the-box availability of many languages, as well as the ease with which high-quality modes may be added for new languages, make it a useful resource for researchers and developers working in the speech and cross-lingual NLP spaces.

8. Acknowledgements

This project was sponsored by the Defense Advanced Research Projects Agency (DARPA) Information Innovation Office (I2O), program: Low Resource Languages for Emergent Incidents (LORELEI), issued by DARPA/I2O under Contract No. HR0011-15-C-0114.

This work used releases IARPA-babel105b-v0.4, IARPA-babel106-v0.2g, IARPA-babel202b-v1.0d, IARPA-babel204b-v1.1b, IARPA-babel205b-v1.0a and IARPA-babel307b-v1.0b provided by IARPA BABEL Research Program

9. Bibliographical References

Bharadwaj, A., Mortensen, D., Dyer, C., and Carbonell, J. G. (2016). Phonologically aware neural model for named entity recognition in low resource transfer settings. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1462–1472, August.

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

Black, A. W. and Lenzo, K. (2001). Flite: a small fast run-time synthesis engine. In ISCA, 4th Speech Synthesis Workshop, pages 157–162, Scotland.

Black, A. W., Lenzo, K., and Pagel, V. (1998). Issues in building general letter to sound rules. In 3rd ESCA Workshop on Speech Synthesis, pages 77–80.

Daelemans, W. and van den Bosch, A., (1996). Language-Independent Data-Oriented Grapheme-to-Phoneme Conversion, pages 77–90. Springer Verlag, New York.

Laurent, A., Deléglise, P., and Meignier, S. (2009). Grapheme to phoneme conversion using an SMT system. In 10th Annual Conference of the International Speech Communication Association 2009 (INTERSPEECH 2009), pages 716–719, Brighton, United Kingdom, September.

Luong, H.-T. and Vu, H.-Q. (2016). A non-expert Kaldi recipe for Vietnamese speech recognition system. In Proceedings of WLSI/OIAF4HLT, Osaka, Japan, December.

Miao, Y., Gowayyed, M., and Metze, F. (2015). EESEN: End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding. ArXiv e-prints, July.

Mortensen, D. R., Littell, P., Bharadwaj, A., Goyal, K., Dyer, C., and Levin, L. (2016). Panphon: A resource for mapping IPA segments to articulatory feature vectors. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3475–3484, Osaka, Japan, December.

Qian, T., Hollingshead, K., Yoon, S., Kim, K., and Sproat, R. (2010). A Python toolkit for universal transliteration. In Nicoletta Calzolari (Conference Chair), et al., editors, Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, may. European Language Resources Association (ELRA).

Rao, K., Peng, F., Sak, H., and Beaufays, F. (2015). Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4225–4229, April.

Sproat, R. (2006). A Computational Theory of Writing Systems. Cambridge University Press, Cambridge.