G2P and P2G

David R. Mortensen

March 21, 2024

Introduction

In this lecture, we will talk about G2P (and a little about its inverse, P2G).

Uses of G2P

G2P used to be more important than it is now. It was an essential part of ASR and TTS systems prior to the end-to-end revolution. Now, G2P is used less consistently in speech, but in a wider diversity of applications. Some examples:

- Low-resource TTS
- Producing IPA transcriptions of recordings of low-resource languages (where recordings are available) so that they can be aligned with the recordings and these can be used, together, to do corpus phonetic research
- As a preprocessing step for NLP tasks that benefit from phonemic representations (like NER, Entity Linking, QA, NLI, etc.)
- Preprocessing for most computational linguistics tasks involving phonology where the input data is in orthography

Uses of P2G

The uses of P2G are more constrained. However, sometimes it is desirable to convert graphemes to phonemes, to apply some processing, and then to convert the phonemes back into graphemes.

Challenges of G2P

There are three significant challenges in G2P:

1. Data is scarce and can be expensive to produce
2. When it does exist, it is usually in the form of transcriptions out of context, rather than running text
3. The grapheme to phoneme mapping is non-deterministic in many languages

WikiPron

One source of pronunciation data is Wiktionary. A tool has been developed for scraping word-transcription pairs from Wiktionary for many languages. It is called WikiPron.[1] Using WikiPron, it is possible to obtain orthography-IPA pairs for a large number of languages, although the quality varies and the number of pairs for a particular language is sometimes small.

Rule-based G2P

Here, we will concentrate on multilingual G2P systems since that is the primary area in which rule-based G2P systems are still used.

Unitran

An early massively multilingual G2P system was called Unitran (ScriptTranscriber). It reflected a mammoth effort of slogging through the entire Unicode standard and providing an IPA equivalent for each grapheme. The Latin alphabet was excluded as too ambiguous.[2] The goal of this tool was not G2P, per se, but transliteration and it was based on very naive views of how orthographies function. However, it was a major step forward toward massively multilingual G2P.

Epitran

To address weakeness in UniTran, Epitran was developed.

[3] is a widely-used G2P system with support for many languages. The current version is written in Python and is distributed as a Python package. Support for English is provided through an external program, `lex_lookup` that implements a WFST-based paired-ngram model. Support Mandarin Chinese and Japanese is provided via pronouncing dictionaries. The other languages are supported through a consistent rule-based architecture.

Conversion from a string of graphemes to a string of IPA phonemes is performed via a three-part pipeline:

preprocessor A cascade of regular expression substitutions (written as string-rewrite rules) that transform the input into a form that can be mapped into IPA.

map A table of string-to-string mappings that is applied to the post-processed string greedily from left to right.

postprocessor Another cascade of regular expression substitutions that transform the IPA string resulting from mapping to the actual pronunciation. This often implements phonological rules that are not reflected in the orthography. For example, the fact that German ⟨n⟩ is realized as /ŋ/ before /k/ does not appear in the writing system, but does appear in speech.

[1] Jackson L. Lee, Lucas F.E. Ashby, M. Elizabeth Garza, Yeonju Lee-Sikka, Sean Miller, Alan Wong, Arya D. McCarthy, and Kyle Gorman. Massively multilingual pronunciation modeling with WikiPron. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4223–4228, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL `https://aclanthology.org/2020.lrec-1.521`

[2] Ting Qian, Kristy Hollingshead, Su-youn Yoon, Kyoung-young Kim, and Richard Sproat. A python toolkit for universal transliteration. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2010/pdf/30_Paper.pdf`

[3] David R. Mortensen, Siddharth Dalmia, and Patrick Littell. Epitran: Precision G2P for many languages. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL `https://aclanthology.org/L18-1429`

The pipeline is illustrated with a German example (*Stecker* 'receptacle') in Figure 1. In preprocessing, ⟨s⟩ is converted to ⟨sch⟩ before ⟨t⟩ so that, when

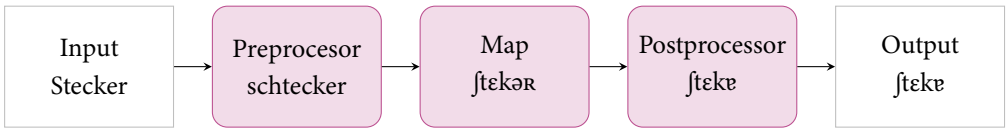

Figure 1: The structure of an Epitran language module

mapping applies, this phoneme (like all other occurrences of ⟨sch⟩) can be mapped to /ʃ/. The map does most of the work, converting each of the characters/substrings to their IPA equivalents. Due to a phonological rule, the sequence /əʀ/ is converted to /ɐ/ (in Standard German but not in many varieties). We have decided to handle this with a postprocessor rule.

How are these rules coded? The first rule, ⟨s⟩ → ⟨sch⟩ / _ t, is written as follows:

```
% Retraction of initial <s> before plosives
s -> sch / # _ (p|t)
```

Lines that begin with % are comments. What this rule means is that ⟨s⟩ is rewritten as ⟨sch⟩ in environment where it is after a word boundary (indicated with #) and before ⟨p⟩ or ⟨t⟩, written as a regex disjunction. Note that this could have also been written as:

```
s -> sch / # _ [pt]
```

since the parts of rules follow the Python regular expression syntax. These rules are placed in a single file and apply in order.

The mapping table is very simple: it is simply a comma-separated value file consisting of grapheme-phoneme pairs:

```
b,b
bb,b
c,k
ch,x
chs,ks
ck,kC
d,d
dd,dC
dsch,d͡ʒ
dt,tC
f,f
ff,fC
g,g
gg,kC
ig,iç
h,h
```

```
j,j
...
```

Note, though, that it is possible to use non-IPA symbols in the output. Here, a generic C is used to stand in for a consonant, which will be used to trigger one or more rules in the postprocessor.

In our discussion of the preprocessor, we left out one important feature: it is possible to define variables that stand in for regular expressions. For example, we might define a variable for consonants:

```
::consonant::␣=␣b|k|x|d|d͡ʒ|t|f|g|ç|j|l|m|n|ŋ|p|p͡f|v|ʁ|s|ʃ|s|t͡ʃ|t͡s|C
```

Note the the variable names must be enclosed in two pairs of colons. It is then possible to use the variable in rules:

```
i -> ɪ / _(::consonant::)(::consonant::)
y -> ʏ / _(::consonant::)(::consonant::)
e -> ɛ / _(::consonant::)(::consonant::)
ø -> œ / _(::consonant::)(::consonant::)
ɑ -> a / _(::consonant::)(::consonant::)
o -> ɔ / _(::consonant::)(::consonant::)
u -> ʊ / _(::consonant::)(::consonant::)
```

(Note that these rules are not quite right, but are provided for illustrative purposes). We are particularly concerned, though, with the rule that converts schwa + /ʁ/ to /ɐ/ at the end of a word. It is, simply:

```
əʁ -> ɐ / _ #
```

Why was Epitran designed so that each language module had these three components (preprocessor, map, postprocessor)? In principle, all three functions could be performed by a single cascade of rewrite rules (or, for that matter, by a single composed series of finite-state transducers). The goal, in this case, was to make development easy. For the majority of languages, only the map is necessary and it is easy to create and edit in a spreadsheet (or using the CSV modes in modern text editors) if it is in CSV format. It is also easy to understand and reason about. The general cases can be handled with the map; the specific cases can be handled either by transforming the orthography using the preprocessor or transforming the phonemes using the postprocessor.

Epitran is currently Python-only (except for `lex_lookup`, which is written in C). Epitran 2, under development, will be written in Rust using WFSTs under the hood (which will make the library faster, will allow for improved handling of exceptional words, and will allow all G2P modules to be inverted into P2G modules). It will also make it easy combine the current rule-based approach with a data-driven approach.

Data-driven G2P

Epitran and similar rule-based models perform very well when the grapheme-to-phoneme relationship is deterministic. In fact, it is not possible for a data-driven G2P system to outperform a correct rule-based G2P. However, as we have seen **the grapheme to phoneme relationship is often not deterministic**. For these purposes, G2P models based on machine learning are needed. These may be neural or non-neural.

Paired-ngram G2P

Early, very successful, G2P models were based on WFSTs (which we used to implement HMMs). The insight, here, is that there is a correspondence between ngrams of graphemes and ngrams of phonemes. For example, ⟨gh⟩ is pronounced three different ways:

- ghost, ghetto
- enough, tough, slough, laugh, trough, rough, cough
- knight, right, fright, night, tight, height, bight, eight, caught, taught, daughter

This seems complicated, but usually when you see, ⟨ough⟩, there is a corresponding /ʌf/ (or sometimes /ɑf/) in the phonemic transcription. Except in ⟨eight⟩, ⟨ight⟩ always corresponds to /ɑɪt/. A paired-ngram G2P is trained on a pronouncing dictionary and learns these associations. The process of doing G2P is that of decoding an HMM where the observations are graphemes and the hidden states correspond to phonemes.

LSTM-based systems

In performance, paired-ngram G2P is generally worse that that of LSTM-based systems (although the LSTM-based models are potentially harder to train and more computationally expensive to use). These models typically use a simple encoder-decoder architecture, or encoder-decoder with some type of attention (hard attention, sparse attention, etc.). Like paired-ngram G2P models, they are typically trained on pronouncing dictionary data (though there is no theoretical reason that they could not be trained on running text, apart from the fact that phonemic transcriptions of running data are scarce).

Transformer-based systems

Transformer-based G2P models perform better than LSTM-based models when data is plentiful. They typically use small transformers (few encoders and decoders). They are computationally much more expensive than LSTMs or WFST-based models but the performance advantage may make them worthwhile.

References

Jackson L. Lee, Lucas F.E. Ashby, M. Elizabeth Garza, Yeonju Lee-Sikka, Sean Miller, Alan Wong, Arya D. McCarthy, and Kyle Gorman. Massively multilingual pronunciation modeling with WikiPron. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4223–4228, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.521.

David R. Mortensen, Siddharth Dalmia, and Patrick Littell. Epitran: Precision G2P for many languages. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1429.

Ting Qian, Kristy Hollingshead, Su-youn Yoon, Kyoung-young Kim, and Richard Sproat. A python toolkit for universal transliteration. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/30_Paper.pdf.