Massively Multilingual Neural Grapheme-to-Phoneme Conversion

Ben Peters
Saarland University
Saarbrücken, Germany
`benzurdopeters@gmail.com`

Jon Dehdari and **Josef van Genabith**
DFKI & Saarland University
Saarbrücken, Germany
`firstname.lastname@dfki.de`

Abstract

Grapheme-to-phoneme conversion (g2p) is necessary for text-to-speech and automatic speech recognition systems. Most g2p systems are monolingual: they require language-specific data or handcrafting of rules. Such systems are difficult to extend to low resource languages, for which data and handcrafted rules are not available. As an alternative, we present a neural sequence-to-sequence approach to g2p which is trained on spelling–pronunciation pairs in hundreds of languages. The system shares a single encoder and decoder across all languages, allowing it to utilize the intrinsic similarities between different writing systems. We show an 11% improvement in phoneme error rate over an approach based on adapting high-resource monolingual g2p models to low-resource languages. Our model is also much more compact relative to previous approaches.

1 Introduction

Accurate grapheme-to-phoneme conversion (g2p) is important for any application that depends on the sometimes inconsistent relationship between spoken and written language. Most prominently, this includes text-to-speech and automatic speech recognition. Most work on g2p has focused on a few languages for which extensive pronunciation data is available (Bisani and Ney, 2008; Novak et al., 2016; Rao et al., 2015; Yao and Zweig, 2015, *inter alia)*. Most languages lack these resources. However, a low resource language's writing system is likely to be similar to the writing systems of languages that do have sufficient pronunciation data. Therefore g2p may be possible for low resource languages if this high resource data

can be properly utilized.

We attempt to leverage high resource data by treating g2p as a multisource neural machine translation (NMT) problem. The source sequences for our system are words in the standard orthography in any language. The target sequences are the corresponding representation in the International Phonetic Alphabet (IPA). Our results show that the parameters learned by the shared encoder–decoder are able to exploit the orthographic and phonemic similarities between the various languages in our data.

2 Related Work

2.1 Low Resource g2p

Our approach is similar in goal to Deri and Knight (2016)'s model for adapting high resource g2p models for low resource languages. They trained weighted finite state transducer (wFST) models on a variety of high resource languages, then transferred those models to low resource languages, using a language distance metric to choose which high resource models to use and a phoneme distance metric to map the high resource language's phonemes to the low resource language's phoneme inventory. These distance metrics are computed based on data from Phoible (Moran et al., 2014) and URIEL (Littell et al., 2017).

Other low resource g2p systems have used a strategy of combining multiple models. Schlippe et al. (2014) trained several data-driven g2p systems on varying quantities of monolingual data and combined their outputs with a phoneme-level voting scheme. This led to improvements over the best-performing single system for small quantities of data in some languages. Jyothi and Hasegawa-Johnson (2017) trained recurrent neural networks for small data sets and found that a version of their system that combined the neural network output

19

with the output of the wFST-based Phonetisaurus system (Novak et al., 2016) did better than either system alone.

A different approach came from Kim and Snyder (2012), who used supervised learning with an undirected graphical model to induce the grapheme–phoneme mappings for languages written in the Latin alphabet. Given a short text in a language, the model predicts the language's orthographic rules. To create phonemic context features from the short text, the model naïvely maps graphemes to IPA symbols written with the same character, and uses the features of these symbols to learn an approximation of the phonotactic constraints of the language. In their experiments, these phonotactic features proved to be more valuable than geographical and genetic features drawn from WALS (Dryer and Haspelmath, 2013).

2.2 Multilingual Neural NLP

In recent years, neural networks have emerged as a common way to use data from several languages in a single system. Google's zero-shot neural machine translation system (Johnson et al., 2016) shares an encoder and decoder across all language pairs. In order to facilitate this multi-way translation, they prepend an artificial token to the beginning of each source sentence at both training and translation time. The token identifies what language the sentence should be translated to. This approach has three benefits: it is far more efficient than building a separate model for each language pair; it allows for translation between languages that share no parallel data; and it improves results on low-resource languages by allowing them to implicitly share parameters with high-resource languages. Our g2p system is inspired by this approach, although it differs in that there is only one target "language", IPA, and the artificial tokens identify the language of the source instead of the language of the target.

Other work has also made use of multilingually-trained neural networks. Phoneme-level polyglot language models (Tsvetkov et al., 2016) train a single model on multiple languages and additionally condition on externally constructed typological data about the language. Östling and Tiedemann (2017) used a similar approach, in which a character-level neural language model is trained on a massively multilingual corpus. A language embedding vector is concatenated to the input at

each time step. The language embeddings their system learned correlate closely to the genetic relationships between languages. However, neither of these models was applied to g2p.

3 Grapheme-to-Phoneme

g2p is the problem of converting the orthographic representation of a word into a phonemic representation. A phoneme is an abstract unit of sound which may have different realizations in different contexts. For example, the English phoneme /p/ has two phonetic realizations (or allophones):

- [pʰ], as in the word 'pain' [pʰ eɪ n]

- [p], as in the word 'Spain' [s p eɪ n]

English speakers without linguistic training often struggle to perceive any difference between these sounds. Writing systems usually do not distinguish between allophones: [pʰ] and [p] are both written as ⟨p⟩ in English. The sounds are written differently in languages where they contrast, such as Hindi and Eastern Armenian.

Most writing systems in use today are glottographic, meaning that their symbols encode solely phonological information[1]. But despite being glottographic, in few writing systems do graphemes correspond one-to-one with phonemes. There are cases in which multiple graphemes represent a single phoneme, as in the word *the* in English:

th   e
ð   ə

There are cases in which a single grapheme represents multiple phonemes, such as syllabaries, in which each symbol represents a syllable.

In many languages, there are silent letters, as in the word *hora* in Spanish:

h   o   r   a
-    o   r   a

There are more complicated correspondences, such as the silent *e* in English that affects the pronunciation of the previous vowel, as seen in the pair of words *cape* and *cap*.

It is possible for an orthographic system to have any or all of the above phenomena while remaining unambiguous. However, some orthographic

---

[1]The Chinese script, in which characters have both phonological form and semantic meaning, is the best-known exception.

systems contain ambiguities. English is well-known for its spelling ambiguities. Abjads, used for Arabic and Hebrew, do not give full representation to vowels.

Consequently, g2p is harder than simply replacing each grapheme symbol with a corresponding phoneme symbol. It is the problem of replacing a grapheme sequence

$$G = g_1, g_2, ..., g_m$$

with a phoneme sequence

$$\Phi = \phi_1, \phi_2, ..., \phi_n$$

where the sequences are not necessarily of the same length. Data-driven g2p is therefore the problem of finding the phoneme sequence that maximizes the likelihood of the grapheme sequence:

$$\hat{\Phi} = \underset{\Phi'}{\arg\max} \Pr(\Phi' \mid G)$$

Data-driven approaches are especially useful for problems in which the rules that govern them are complex and difficult to engineer by hand. g2p for languages with ambiguous orthographies is such a problem. Multilingual g2p, in which the various languages have similar but different and possibly contradictory spelling rules, can be seen as an extreme case of that. Therefore, a data-driven sequence-to-sequence model is a natural choice.

4 Methods

4.1 Encoder–Decoder Models

In order to find the best phoneme sequence, we use a neural encoder–decoder model with attention (Bahdanau et al., 2014). The model consists of two main parts: the **encoder** compresses each source grapheme sequence G into a fixed-length vector. The **decoder**, conditioned on this fixed-length vector, generates the output phoneme sequence Φ.

The encoder and decoder are both implemented as recurrent neural networks, which have the advantage of being able to process sequences of arbitrary length and use long histories efficiently. They are trained jointly to minimize cross-entropy on the training data. We had our best results when using a bidirectional encoder, which consists of two separate encoders which process the input

Enc. & dec. model type	LSTM
Attention	General
Enc. & dec. layers	2
Hidden layer size	150
Source embedding size	150
Target embedding size	150
Batch size	64
Optimizer	SGD
Learning rate	1.0
Training epochs	13

Table 1: Hyperparameters for multilingual g2p models

in forward and reverse directions. We used long short-term memory units (Hochreiter and Schmidhuber, 1997) for both the encoder and decoder. For the attention mechanism, we used the general global attention architecture described by Luong et al. (2015).

We implemented[2] all models with OpenNMT (Klein et al., 2017). Our hyperparameters, which we determined by experimentation, are listed in Table 1.

4.2 Training Multilingual Models

Presenting pronunciation data in several languages to the network might create problems because different languages have different pronunciation patterns. For example, the string 'real' is pronounced differently in English, German, Spanish, and Portuguese. We solve this problem by prepending each grapheme sequence with an artificial token consisting of the language's ISO 639-3 code enclosed in angle brackets. The English word 'real', for example, would be presented to the system as

```
<eng> r e a l
```

The artificial token is treated simply as an element of the grapheme sequence. This is similar to the approach taken by Johnson et al. (2016) in their zero-shot NMT system. However, their source-side artificial tokens identify the target language, whereas ours identify the source language. An alternative approach, used by Östling and Tiedemann (2017), would be to concatenate a language embedding to the input at each time step. They do not evaluate their approach on grapheme-to-phoneme conversion.

5 Data

In order to train a neural g2p system, one needs a large quantity of pronunciation data. A standard

---

dataset for g2p is the Carnegie Mellon Pronouncing Dictionary (Lenzo, 2007). However, that is a monolingual English resource, so it is unsuitable for our multilingual task. Instead, we use the multilingual pronunciation corpus[3] collected by Deri and Knight (2016) for all experiments. This corpus consists of spelling–pronunciation pairs extracted from Wiktionary. It is already partitioned into training and test sets. Corpus statistics are presented in Table 2.

In addition to the raw IPA transcriptions extracted from Wiktionary, the corpus provides an automatically cleaned version of transcriptions. Cleaning is a necessary step because web-scraped data is often noisy and may be transcribed at an inconsistent level of detail. The data cleaning used here attempts to make the transcriptions consistent with the phonemic inventories used in Phoible (Moran et al., 2014). When a transcription contains a phoneme that is not in its language's inventory in Phoible, that phoneme is replaced by the phoneme with the most similar articulatory features that is in the language's inventory. Sometimes this cleaning algorithm works well: in the German examples in Table 3, the raw German symbols /χ/ and /ç/ are both converted to /x/. This is useful because the /χ/ in *Ansbach* and the /ç/ in *Kaninchen* are instances of the same phoneme, so their phonemic representations should use the same symbol. However, the cleaning algorithm can also have negative effects on the data quality. For example, the phoneme /ɪ/ is not present in the Phoible inventory for German, but it *is* used in several German transcriptions in the corpus. The cleaning algorithm converts /ɪ/ to /l/ in all German transcriptions, whereas /r/ would be a more reasonable guess. The cleaning algorithm also removes most suprasegmentals, even though these are often an important part of a language's phonology. Developing a more sophisticated procedure for cleaning pronunciation data is a direction for future work, but in this paper we use the corpus's provided cleaned transcriptions in order to ease comparison to previous results.

6 Experiments

We present experiments with two versions of our sequence-to-sequence model. LangID prepends each training, validation, and test sample with

Split	Train	Test
Languages	311	507
Words	631,828	25,894
Scripts	42	45

Table 2: Corpus Statistics

Lang.	Script	Spelling	Cleaned IPA	Raw IPA
deu	Latin	Ansbach	aː n s b aː x	ˈansbaχ
deu	Latin	Kaninchen	k aː n ɪ n x ə n	kaˈniːnçən
eus	Latin	untxi	u n̪ t̪ ʃ ɪ	ˈun.t͡ʃi

Table 3: Example entries from the Wiktionary training corpus

an artificial token identifying the language of the sample. NoLangID omits this token. LangID and NoLangID have identical structure otherwise. To translate the test corpus, we used a beam width of 100. Although this is an unusually wide beam and had negligible performance effects, it was necessary to compute our error metrics.

6.1 Evaluation

We use the following three evaluation metrics:

- Phoneme Error Rate (PER) is the Levenshtein distance between the predicted phoneme sequences and the gold standard phoneme sequences, divided by the length of the gold standard phoneme sequences.

- Word Error Rate (WER) is the percentage of words in which the predicted phoneme sequence does not exactly match the gold standard phoneme sequence.

- Word Error Rate 100 (WER 100) is the percentage of words in the test set for which the correct guess is not in the first 100 guesses of the system.

In system evaluations, WER, WER 100, and PER numbers presented for multiple languages are averaged, weighting each language equally (following Deri and Knight, 2016).

It would be interesting to compute error metrics that incorporate phoneme similarity, such as those proposed by Hixon et al. (2011). PER weights all phoneme errors the same, even though some errors are more harmful than others: /d/ and /k/ are usually contrastive, whereas /d/ and /d̪/ almost never are. Such statistics would be especially interesting for evaluating a multilingual system, because

different languages often map the same grapheme to phonemes that are only subtly different from each other. However, these statistics have not been widely reported for other g2p systems, so we omit them here.

6.2 Baseline

Results on LangID and NoLangID are compared to the system presented by Deri and Knight (2016), which is identified in our results as wFST. Their results can be divided into two parts:

- High resource results, computed with wFSTs trained on a combination of Wiktionary pronunciation data and g2p rules extracted from Wikipedia IPA Help pages. They report high resource results for 85 languages.

- Adapted results, where they apply various mapping strategies in order to adapt high resource models to other languages. The final adapted results they reported include most of the 85 languages with high resource results, as well as the various languages they were able to adapt them for, for a total of 229 languages. This test set omits 23 of the high resource languages that are written in unique scripts or for which language distance metrics could not be computed.

6.3 Training

We train the LangID and NoLangID versions of our model each on three subsets of the Wiktionary data:

- LangID-High and NoLangID-High: Trained on data from the 85 languages for which Deri and Knight (2016) used non-adapted wFST models.

- LangID-Adapted and NoLangID-Adapted: Trained on data from any of the 229 languages for which they built adapted models. Because many of these languages had no training data at all, the model is actually only trained on data in 157 languages. As is noted above, the Adapted set omits 23 languages which are in the High test set.

- LangID-All and NoLangID-All: Trained on data in all 311 languages in the Wiktionary training corpus.

In order to ease comparison to Deri and Knight's system, we limited our use of the training corpus to 10,000 words per language. We set aside 10 percent of the data in each language for validation, so the maximum number of training words for any language is 9000 for our systems.

6.4 Adapted Results

On the 229 languages for which Deri and Knight (2016) presented their final results, the LangID version of our system outperforms the baseline by a wide margin. The best performance came with the version of our model that was trained on data in all available languages, not just the languages it was tested on. Using a language ID token improves results considerably, but even NoLangID beats the baseline in WER and WER 100. Full results are presented in Table 4.

Model	WER	WER 100	PER
wFST	88.04	69.80	48.01
LangID-High	74.99	46.18	42.64
LangID-Adapted	75.06	46.39	41.77
LangID-All	**74.10**	**43.23**	**37.85**
NoLangID-High	82.14	50.17	54.05
NoLangID-Adapted	85.11	48.24	55.93
NoLangID-All	83.65	47.13	51.87

Table 4: Adapted Results

6.5 High Resource Results

Having shown that our model exceeds the performance of the wFST-adaptation approach, we next compare it to the baseline models for just high resource languages. The wFST models here are purely monolingual – they do not use data adaptation because there is sufficient training data for each of them. Full results are presented in Table 5. We omit models trained on the Adapted languages because they were not trained on high resource languages with unique writing systems, such as Georgian and Greek, and consequently performed very poorly on them.

In contrast to the larger-scale Adapted results, in the High Resource experiments none of the sequence-to-sequence approaches equal the performance of the wFST model in WER and PER, although LangID-High does come close. The LangID models do beat wFST in WER 100. A possible explanation is that a monolingual wFST

model will never generate phonemes that are not part of the language's inventory. A multilingual model, on the other hand, could potentially generate phonemes from the inventories of any language it has been trained on.

Even if LangID-High does not present a more accurate result, it does present a more compact one: LangID-High is 15.4 MB, while the combined wFST high resource models are 197.5 MB.

Model	WER	WER 100	PER
wFST	**44.17**	21.97	**14.70**
LangID-High	47.88	**15.50**	16.89
LangID-All	48.76	15.78	17.35
NoLangID-High	69.72	29.24	35.16
NoLangID-All	69.82	29.27	35.47

Table 5: High Resource Results

6.6 Results on Unseen Languages

Finally, we report our models' results on unseen languages in Table 6. The unseen languages are any that are present in the test corpus but absent from the training data. Deri and Knight did not report results specifically on these languages. Although the NoLangID models sometimes do better on WER 100, even here the LangID models have a slight advantage in WER and PER. This is somewhat surprising because the LangID models have not learned embeddings for the language ID tokens of unseen languages. Perhaps negative associations are also being learned, driving the model towards predicting more common pronunciations for unseen languages.

Model	WER	WER 100	PER
LangID-High	**85.94**	58.10	**53.06**
LangID-Adapted	87.78	68.40	65.62
LangID-All	86.27	62.31	54.33
NoLangID-High	88.52	58.21	62.02
NoLangID-Adapted	91.27	57.61	74.07
NoLangID-All	89.96	**56.29**	62.79

Table 6: Results on languages not in the training corpus

7 Discussion

7.1 Language ID Tokens

Adding a language ID token always improves results in cases where an embedding has been learned for that token. The power of these embeddings is demonstrated by what happens when one feeds the same input word to the model with different language tokens, as is seen in Table 7. Impressively, this even works when the source sequence is in the wrong script for the language, as is seen in the entry for Arabic.

Language	Pronunciation
English	d ʒ uː æɪ s
German	j ʊ t s ə
Spanish	x w i θ ẹ
Italian	d ʒ u i t ʃ e
Portuguese	ʒ w i s ĩ
Turkish	ʒ ʊ ɪ d̩ ʒ ɛ
Arabic	j uː i s

Table 7: The word 'juice' translated by the LangID-All model with various language ID tokens. The incorrect English pronunciation rhymes with the system's result for 'ice'

7.2 Language Embeddings

Because these language ID tokens are so useful, it would be good if they could be effectively estimated for unseen languages. Östling and Tiedemann (2017) found that the language vectors their models learned correlated well to genetic relationships, so it would be interesting to see if the embeddings our source encoder learned for the language ID tokens showed anything similar. In a few cases they do (the languages closest to German in the vector space are Luxembourgish, Bavarian, and Yiddish, all close relatives). However, for the most part the structure of these vectors is not interpretable. Therefore, it would be difficult to estimate the embedding for an unseen language, or to "borrow" the language ID token of a similar language. A more promising way forward is to find a model that uses an externally constructed typological representation of the language.

7.3 Phoneme Embeddings

In contrast to the language embeddings, the phoneme embeddings appear to show many regularities (see Table 8). This is a sign that our multilingual model learns similar embeddings for

phonemes that are written with the same grapheme in different languages. These phonemes tend to be phonetically similar to each other.

Perhaps the structure of the phoneme embedding space is what leads to our models' very good performance on WER 100. Even when the model's first predicted pronunciation is not correct, it tends to assign more probability mass to guesses that are more similar to the correct one. Applying some sort of filtering or reranking of the system output might therefore lead to better performance.

Phoneme	Closest phonemes
b	p^h, β, ɸ
ə	ã, ĕ, ɯ
t^h	tː, t, t̪
x	χ, ɣ, ħ
y	yː, ʏ, ɪ
ɹ	ɾ^ɣ, r̪, ɾ

Table 8: Selected phonemes and the most similar phonemes, measured by the cosine similarity of the embeddings learned by the LangID-All model

7.4 Future Work

Because the language ID token is so beneficial to performance, it would be very interesting to find ways to extend a similar benefit to unseen languages. One possible way to do so is with tokens that identify something other than the language, such as typological features about the language's phonemic inventory. This could enable better sharing of resources among languages. Such typological knowledge is readily available in databases like Phoible and WALS for a wide variety of languages. It would be interesting to explore if any of these features is a good predictor of a language's orthographic rules.

It would also be interesting to apply the artificial token approach to other problems besides multilingual g2p. One closely related application is monolingual English g2p. Some of the ambiguity of English spelling is due to the wide variety of loanwords in the language, many of which have unassimilated spellings. Knowing the origins of these loanwords could provide a useful hint for figuring out their pronunciations. The etymology of a word could be tagged in an analogous way to how language ID is tagged in multilingual g2p.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv preprint* 1409.0473. http://arxiv.org/abs/1409.0473.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication* 50(5):434–451.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-phoneme models for (almost) any language. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 1, pages 399–408.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. http://wals.info/.

Ben Hixon, Eric Schneider, and Susan L Epstein. 2011. Phonemic similarity metrics to compare pronunciation methods. In *Twelfth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Florence, Italy, pages 825–828.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *ArXiv preprint* 1611.04558. http://arxiv.org/abs/1611.04558.

Preethi Jyothi and Mark Hasegawa-Johnson. 2017. Low-resource grapheme-to-phoneme conversion using recurrent neural networks. In *Proc. ICASSP*.

Young-Bum Kim and Benjamin Snyder. 2012. Universal grapheme-to-phoneme prediction over Latin alphabets. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 332–343.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv preprint* https://arxiv.org/abs/1701.02810.

Kevin Lenzo. 2007. The CMU pronouncing dictionary.

Patrick Littell, David Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *EACL 2017*. Valencia, Spain, pages 8–14.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR* abs/1508.04025. http://arxiv.org/abs/1508.04025.

Steven Moran, Daniel McCloy, and Richard Wright, editors. 2014. *PHOIBLE Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. http://phoible.org .

Josef Robert Novak, Nobuaki Minematsu, and Keikichi Hirose. 2016. Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint *n*-gram models in the WFST framework. *Natural Language Engineering* 22(6):907–938.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. *EACL 2017* page 644.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 4225–4229.

Tim Schlippe, Wolf Quaschningk, and Tanja Schultz. 2014. Combining grapheme-to-phoneme converter outputs for enhanced pronunciation generation in low-resource scenarios. In *International Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU)*. pages 139–145.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *NAACL 2016*. San Diego, California, pages 1357–1366.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *ArXiv preprint* 1506.00196. http://arxiv.org/abs/1506.00196.