Zero-shot Learning for Grapheme to Phoneme Conversion with Language Ensemble

Anonymous ACL submission

Abstract

Grapheme-to-Phoneme (G2P) has many applications in NLP and speech fields. Most existing work focuses heavily on languages with abundant training datasets, which limit the scope of target languages to less than 100 languages. This work attempts to apply zero-shot learning to propose G2P models for all low-resource and endangered languages in Glottolog (about 8k languages). For any unseen target language, we first build the phylogenetic tree (i.e. language family tree) to identify top-k nearest languages for which we have training sets. Then we run models of those languages to obtain a hypothesis set, which we combine into a confusion network to propose a most likely hypothesis as an approximation to the target language. We test our approach on over 600 unseen languages and demonstrate it significantly outperforms baselines.

1 Introduction

Grapheme-to-Phoneme (G2P) plays a crucial role in many NLP tasks. In particular, it is used heavily in many speech-related tasks such as speech recognition and speech synthesis (Arık et al., 2017; Miao et al., 2015). Even in the latest end-to-end systems, it still has a strong impact on the speech performance (Hayashi et al., 2021). Typically, the G2P task is language-dependent—many language-specific factors affect the G2P process such as the general characteristics of scripts (Ager, 2008), phonotactic constraints (Hayes and Wilson, 2008) and other orthography factors (Frost and Katz, 1992). Therefore, to develop a G2P model, we need either to create a training set for the target language, like (CMU, 2000), or to ask linguists to explicitly define a set of orthographic rules to map from graphemes to phonemes (Mortensen et al., 2018). Both approaches have achieved success for high-resource languages; however, they can only account for a small number of the world's languages. The majority still do not have access to G2P due to limited training resources.

In this work, we attempt to solve this problem by proposing zero-shot learning for G2P using the language ensemble approach. Our approach allows us to cover all languages present in the GlottoLog database: around 8000 of them (Nordhoff and Hammarström, 2011). The main insight of our approach is that we can approximate the G2P model of an unseen language using those of related languages because languages related to the target language should have similar orthographic rules (of both the context-free and context-dependent type). For example, a native speaker of Italian (a Romance language) is likely to make accurate guesses about how a text in Catalan (another Romance language) would be pronounced. We define the similarity between languages as the shortest distance between two languages in the phylogenetic tree (i.e. language family tree). We first build models for the subset of languages (training languages) where we have a large enough training set (e.g., Italian, Spanish, etc.). Then, for each unseen language (e.g., Catalan), we first find the top-k nearest training languages (like Italian, Spanish, etc.) and use those languages' G2P models to generate k hypotheses. Finally, we ensemble the G2P outputs by building a confusion network and discover the most-likely sequence as an approximation to the target language.

In our experiments, we build a large dataset from Wiktionary in which we use 260 languages as the training languages and test our approach on 600 unseen languages. We apply our approach to 3 different architectures: a joint-sequence n-gram model (Novak et al., 2016), an LSTM sequence-to-sequence model (Rao et al., 2015), and a transformer sequence-to-sequence model (Peters et al., 2017). Using any of the architectures, our approach outperforms all baseline by more than 5% PER.

The main contributions of this work are as follows:

1. A novel approach to approximate target language G2P models using the nearest languages in a phylogenetic tree

2. An approach to ensemble predictions from multiple outputs using confusion networks.

3. A demonstration that our approach achieves significantly better performance than baselines when testing on 600 unseen languages.

2 Related Work

Traditionally, a G2P component is built using rule-based models. For example, the phonological constraints can be incorporated into context-sensitive grammars and implemented using finite-state transducers (Kaplan and Kay, 1994). However, designing the rules requires many hours from linguists and can be prohibitive for low-resource languages if they have deep orthographies[1].

Statistical models overcome this problem by learning the rules automatically. Typically, there are two steps in building such a model: first, the sequence of phonemes and graphemes are aligned to each other, then another prediction model is built on top of the alignment. The alignment model is typically done using Expectation and Maximization (Ristad and Yianilos, 1998; Jiampojamarn and Kondrak, 2010). The prediction model can be done using neural networks (Sejnowski and Rosenberg, 1987), decision trees (Black et al., 1998; Daelemans and Van den Bosch, 1997), joint-sequence models (Bisani and Ney, 2008) and WFST-based n-gram models (Novak et al., 2016). More recently, deep neural networks have been applied to the G2P task. Various architectures have been explored, for example, RNNs (Rao et al., 2015; Yao and Zweig, 2015; Lee et al., 2020), CNNs (Yolchuyeva et al., 2019) and Transformers (Yolchuyeva et al., 2020).

Traditionally, each G2P model was typically built for one high-resource language. Recently, many researchers have started to focus on low-resource G2P models. One related work adapts high-resource language models to low-resource language models by measuring similarity between languages and phonemes (Deri and Knight, 2016). This previous work creates a new training set for every low-resource language by adapting the training set from the top-3 nearest languages. However, there are several issues with this approach. First, it has to prepare separate training sets and n-gram models for every testing language, which is quite computationally expensive. It also suffers from the limited training set problem even after merging top-3 languages because most training languages have less than 100 vocabularies, which is insufficient to train any stable neural models. In contrast, we only prepare one unified training set and one unified model in our neural approach, which circumvents these problems. Additionally, the testing languages and training languages are mixed in this work, therefore the performance on unseen languages is not clear. Only a limited number of papers so far focus on developing G2P models for unseen languages. The most common strategy is to drop the target language information and make predictions using a shared multilingual model (Peters et al., 2017; Bleyan et al., 2019). This is one of our baseline (the global language model) in this work.

3 Approach

In this section, we describe our zero-shot learning approach. We first introduce three G2P models to be used for supervised learning and covering high-resource languages. Next, we define the language similarity and language families. Finally, we explain how to ensemble nearest languages models to predict G2P for an unseen language.

3.1 Monolingual Model

In this section, we introduce our monolingual G2P models: a joint n-gram model based on WFSTs, two neural models based on sequence-to-sequence LSTMs, and transformer models. We select those models as they are the three baseline models used in the SIGMORPHON Multilingual G2P task (Gorman et al., 2020). These models are trained for every training language and then used as building blocks to approximate G2P models for unseen testing languages.

The joint n-gram model is a standard monolingual G2P model (Novak et al., 2016). For each training language, the dataset is first aligned using Expectation Maximization, then an n-gram model is built using a WFST[2]. The neural model is a standard sequence to sequence model. We tried two common architectures: bidirectional LSTM and transformer. Unlike the n-gram model, the

[1] Orthographies in which the relationship between graphemes and phonemes has been obscured by history or is otherwise complicated.

[2] https://github.com/AdolfVonKleist/Phonetisaurus

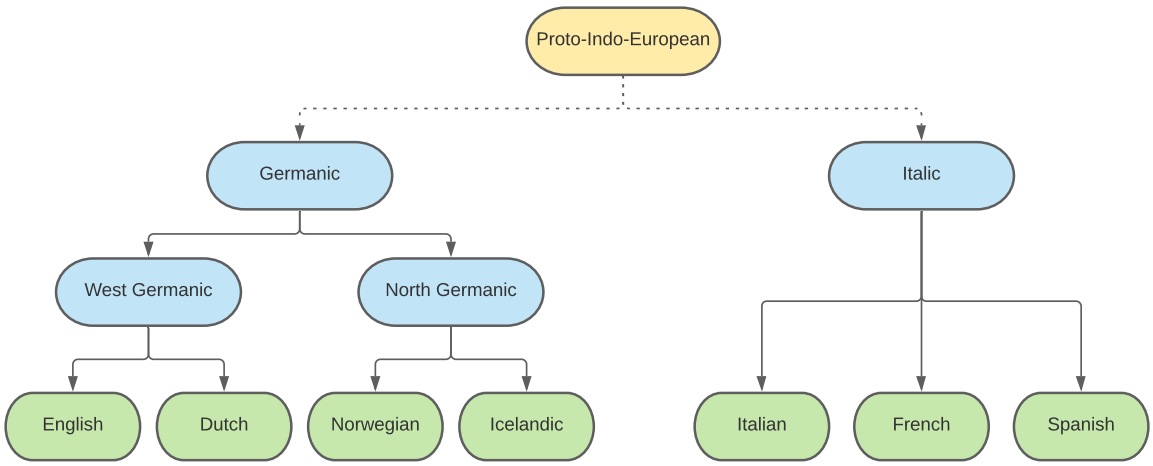

Figure 1: Illustration of a partial phylogenetic tree (i.e. language family tree). The subtree has Proto-Indo-European as the root of the family (there also exists many other root language families). The Germanic branch and Italic branch can be derived (not directly though) from the Proto-Indo-European, they are further divided into the modern languages we are using today. This information can help us compute the similarity between languages.

neural model is trained by combining all training sets into one large dataset. To distinguish different languages, a ISO 639-3 language ID is attached to the input sequence, for example, we attach the "<eng>" to "hello", so the input sequence is "<eng> h e l l o". This approach was explored in previous work (Peters et al., 2017). It allows the parameters to be shared across different languages. Even language with a limited training set could benefit from other high-resource languages.

3.2 Phylogenetic Tree and Nearest Languages

The model discussed in the previous subsection could predict phonemes for any training language, however, it cannot deal with any unseen languages. Our main contribution in this work is to select the highly related languages and then effectively combine those models to approximate the target language. In this subsection, we introduce the concept of the nearest language in terms of the phylogenetic tree (i.e. language family tree), then we explain how we ensemble nearest languages.

There are many metrics to measure the distance between languages from different perspectives (Dryer and Haspelmath, 2013; Littell et al., 2017). In this work, we only consider the phylogenetic tree (i.e., language family tree) to measure the distance between languages. This is because the phylogenetic information is available for a larger portion of languages than any of the other bases of linguistic distance or similarity. Glottolog provides us with language family information for around 8000 languages (Nordhoff and Hammarström, 2011).

In Figure.1, we write a subtree of the entire phylogenetic tree, in particular, it illustrates two major branches of the linguistic *Stammbaum*: the Germanic and Italic. Both of them are children of the Proto-Indo-European (PIE) node. The tree also indicates that English and Dutch are closely related languages and that Norwegian and Icelandic are closely related languages. To measure the distance between any pair of languages, we can compute the length of the shortest path between the two languages. In our example, the English/Dutch pair has a distance 2, and the English/Norwegian pair has a distance of 4. The shortest path can be computed efficiently by using Lowest Common Ancestor (LCA).

$$d(l_1, l_2) = H(l_1) + H(l_2) - H(LCA(l_1, l_2)) \quad (1)$$

where $d(l_1, l_2)$ is the distance between language l_1 and l_2, H compute the height of a node in the tree. This time complexity is $O(log(M))$ where M is the max height of the phylogenetic tree (Cormen et al., 2009). Suppose the entire language set is L and training languages are $T \subset L$, we could compute the k nearest languages for every language $l \in L$, those languages would allow us to ensemble models. Note that it is not always the case that languages belonging to the same family should

share similar orthography. They are also influenced by non-linguistic aspects such as political factors and cultural factors. Additionally, some languages are written in multiple scripts. For example, Uzkek is written with a Perso-Arabic, Cyrillic, and Latin script. However, information on language families provides a reasonable starting point.

3.3 Model Ensemble

After obtaining the nearest languages and the monolingual model for each of the training languages, we can use those models to approximate the target model. In particular, we are interested in combining prediction outputs from different models to create a single prediction output. If the models are one of the local prediction models (i.e: for each grapheme, we decide whether to generate a phoneme and which phoneme to generate) (Sejnowski and Rosenberg, 1987; Black et al., 1998), the ensemble task is simple. As we made one phoneme prediction at every grapheme position, we can use the voting to decide the most likely phoneme.

$$[\hat{p}] = \text{argmax}_{[p]} \sum_i \mathbf{1}([p] = [p]_i) \qquad (2)$$

However, for the more general sequence-to-sequence neural model, it is much complicated. Different models would predict outputs with variable sequences, therefore voting at each position would be meaningless. For example, suppose two phoneme sequences "/helo/" and "/elo/" are generated from "hello" using two different languages. It is difficult to average /h/ and /e/ as they are corresponding to different graphemes. To solve this problem, we use a robust approach to ensemble outputs with variable lengths. Our approach is similar to the ROVER system (Fiscus, 1997), which is a commonly used approach to combine multiple speech outputs into one output. It has been applied to combine phoneme sequence (Schlippe et al., 2014), but only under the monolingual scenario in which they combine different models to improve the performance. This work focus on combining multilingual outputs and modifying the standard word-based network to consider the phonological structure.

One actual example from our dataset is illustrated in Figure.2. First, we build one *confusion network* (or *lattice*) per language in our nearest language set. The raw confusion network represents a single hypothesis using a directed graph whose edge corresponds to a single phoneme from the hypothesis[3]. When we compose multiple confusion networks into one confusion network, there would typically be more than one edge connecting two nodes. The set of edges connecting two contiguous nodes is typically referred to as the *confusion set* (or *correspondence set*) (Fiscus, 1997; Mangu et al., 2000). For example, the first confusion set from the right network in Figure.2 is $\{/t/, /s/\}$. The goal of our ensemble approach is to compose all confusion networks into a single network, and then pick up the best hypothesis from the composed network.

Unlike the original work in which hypotheses are composed without any specific order, we iteratively compose the network using the nearest order: we first compose the nearest and second nearest confusion network into a single network, then further merge the third nearest network into it. In each composition step, we align two networks by computing the similarity between pairs of confusion sets. While the standard network computes the similarity step using the exact matching metric, we relax this exact matching scheme and use a more coarse matching strategy by considering the phonological distance structure. In particular, we use the *phonologically-equivalent class*, which collapses similar sounds into a small number of classes (Mortensen et al., 2016). This means we could easier match $/a/, /o/$ (vowel pairs) than $/a/, /s/$ (vowel, consonant pairs). After composing all confusion networks into one network, the most likely phoneme sequence can be generated from the final network. To generate the sequence, we pick up 1 phoneme per confusion set and concatenate them together. The phoneme in each confusion set is selected using the voting scheme. When there are multiple candidates with equal votes, we break the tie by selecting the candidate generated from the nearest language. Algorithm 1 summarizes the entire steps in our approach.

4 Experiments

In this section, we show the experiment results on our G2P models. First, we introduce the main datasets we used to build our model, next we describe our baseline models and G2P architectures we use in our experiments. Finally, we demonstrate

[3] We can also generate n-best hypotheses from each model and build confusion networks, however, we only consider the top-1 hypothesis in this work for simplicity. N-best hypotheses might be a future work

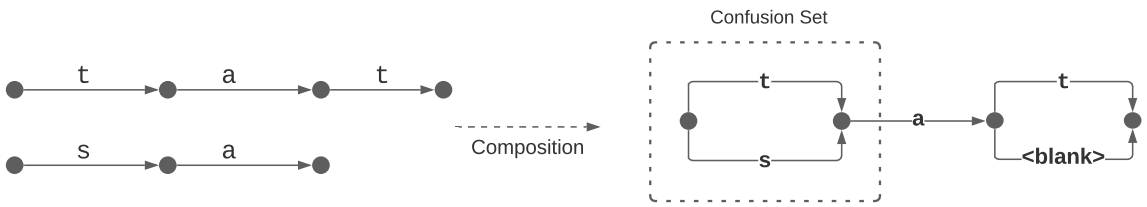

Figure 2: An illustration of an actual ensemble example from our dataset. The input is 'that' from Old Dutch (odt), its top-2 nearest language in our training set are Dutch (nld) and Middle Dutch (dum). The left-hand side denotes two hypotheses generated from those two languages, from which we compose into a confusion network. The composed confusion network has three confusion sets, which would vote '/t a t/' as a final prediction.

Algorithm 1: G2P algorithm

Data: $input, lang$ (Grapheme sequence and its language)
Result: $output$ (ensembled phoneme sequence)
$klangs \leftarrow KNearestLanguage(lang)$
$hyps \leftarrow []$
for $klang \in klangs$ **do**
  $hyp \leftarrow G2P(input, klang)$;
  /* Generate hypothesis for every nearest language */
  $hyps.append(hyp)$
end
$x \leftarrow ConfusionNetwork()$
for $hyp \in hyps$ **do**
  $n \leftarrow ConfusionNetwork(hyp)$
  $a \leftarrow align(x, n)$
  $x \leftarrow composite(x, n, a)$
end
$output \leftarrow []$
for $cs \in x$ **do**
  $p \leftarrow vote(cs)$; /* vote 1 phoneme per confusion set */
  $output.append(p)$
end

that the proposed ensemble approach outperforms those baseline models in different architectures.

4.1 Data

The main training/testing dataset we used is the Wiktionary website. Wiktionary is a large multilingual website containing lexicon information for many languages, including many low-resource languages. One previous work has prepared a dataset using Wiktionary (Deri and Knight, 2016), but the testing languages and training languages are mixed together in this dataset: many testing languages are also available as training languages. To demonstrate our approach on unseen languages, we create a new dataset using the latest Wiktionary. First, we download a dump file from the website and extract all words with pronunciation information[4]. We group all words by their languages, which gives us 972 languages in total. However, not all languages yield a similar number of training data. Figure.3 shows the log-scaled histogram of language counts for different vocabulary sizes: only 1 language: English, has more than 400k vocabulary items. Most of the languages are concentrated in the lowest histogram bar. In our dataset, we find that the majority of the language have less than 100 items. Therefore, the model needs to be able to handle low-resource training scenarios.

Next, most languages from Wiktionary can be assigned an ISO 639-3 ID, which can be identified in our phylogenetic tree. As mentioned in the previous section, our phylogenetic tree is built using the Glottolog database (Nordhoff and Hammarström, 2011), which contains phylogenetic information about 7915 languages. We split all languages into training languages or testing languages depending on the vocabulary size: we consider the language to be a training language if the vocabulary size is above a predefined threshold, otherwise, it is classified as a testing language. Typically, there is a trade-off when selecting the threshold: making the threshold lower would increase the number of training languages and make it easier to find the nearest languages, however lower threshold make the training process more difficult due to the number of

[4] https://github.com/tatuylonen/wiktextract

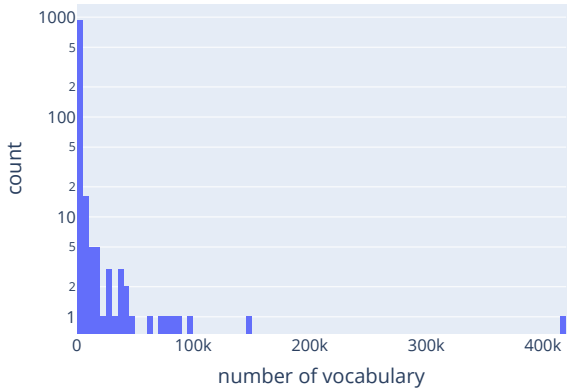

Figure 3: Log-scaled histograms of the count of languages grouped by the number of vocabulary items available in Wiktionary. The language with over 400k vocabulary is English, however, most languages are low-resource languages for which we have less that 100 Wiktionary entries.

Dataset	# Languages	# Vocabulary
Training set	269	1,672,444
Testing set	605	4,796
All	874	1,677,240

Table 1: Statistics of the Wiktionary dataset we used in the experiment. 269 languages are used for training and 605 languages are used for testing.

limited vocabulary, additionally, it would reduce the number of testing languages. In our experiment, the threshold is set to 50 by following the previous work (Deri and Knight, 2016), and the statistics of both training datasets and test datasets are shown in Table.1. We have 269 training languages and 605 testing languages. Most of the training languages have a large number of vocabulary items but the testing languages have only 8 vocabulary items per language. We train both the n-gram model and neural models using only the training languages, and then test them on the testing languages, which are not seen during the training process. The evaluation is done using the average PER (phoneme error rate) across all testing languages.

4.2 Baselines

In our experiments, we consider three different baseline models: the **fixed language model**, which is a model trained using the English dataset. The **global language model** is a shared model mixing all training sets, it ignores the target language id during inference, this was explored in the previous work (Peters et al., 2017). The **nearest language model** can be seen as a special case of our proposed model: we compute the most similar language to the target language and run inference using that language's model instead. For each of the baseline models, we investigate three different architectures: N-gram, LSTM, and transformer architecture. We use OpenNMT-py[5] for our neural models. The LSTM architecture is using the framework's default configuration: 2 standard LSTM layers for both encoder and an attention-based decoder, each layer has 500 hidden size. This model is optimized with 1.0 learning rate using SGD optimizer. The transformer model uses the framework's WMT sample configuration[6]: we have 6 layers for both the encoder and decoder with 500 attention and feedforward size. The mode has a positional encoding layer and is using 8 heads in self-attention. The optimizer is Adam with learning rate 2.0 and 8000 steps for warmup. Both neural models are trained with 20k steps. In our ensemble model, we use the top-10 languages ($k = 10$) in our main experiment.

4.3 Results

Table.2 shows our experiment results. For each of the G2P architecture (N-gram Model, LSTM Model, Transformer Model), we demonstrate our ensemble model's results as well as 3 baselines. The leftmost architecture shows the N-gram Model result: the fixed language model performs 76% PER, The global language model get 70%, which is better than the fixed language model. the nearest language model further improves it to 68%. While all those models perform poorly, the reason for their poor performance is different from each other: the fixed language model is only trained with the English dataset, therefore it cannot handle orthography rules in other languages. The global language model suffers from the inconsistency of the training set: the same grapheme might map to different phonemes in different languages, therefore it cannot learn consistent rules across all languages. Finally, the nearest language model has the problem that the nearest language might be a low-resource language. As we mention in the previous section, most languages have few training vocabularies, even we restrict the training lan-

[5] https://github.com/OpenNMT/OpenNMT-py
[6] https://opennmt.net/OpenNMT-py/FAQ.html#how-do-i-use-the-transformer-model

	N-gram Model				LSTM Model				Transformer Model			
	PER	Add	Del	Sub	PER	Add	Del	Sub	PER	Add	Del	Sub
Fixed Model	76.0	4.52	9.39	62.1	78.1	4.53	20.4	53.2	78.5	3.2	19.0	56.2
Global Model	70.4	6.89	9.86	53.6	72.8	3.4	29.0	43.4	74.2	2.9	20.6	50.8
Nearest Model	68.4	4.51	12.4	51.5	43.8	12.1	4.0	27.6	45.4	15.8	3.6	26.1
Ensemble Model	**55.0**	0.56	23.6	30.9	**35.7**	10.0	3.4	22.2	**39.8**	13.9	3.1	22.8

Table 2: Experiment Results of the our approach. It compares our ensemble model with three baselines: Fixed Model, Global Model and Nearest Model. The comparison is performed under three different architectures: N-gram model, LSTM model, Transformer Model. In all settings, the proposed model outperforms baselines.

guages to have more than 50 vocabularies, the large proportion of languages still have 50 to 100 vocabularies, which might be insufficient to train a good model. Additionally, depending on a single language might have a large variance. The proposed ensemble model solves those issues to some extent: it relies on more than 1 language when predicting for the target language: even 1 language is a low-resource language, other languages might be able to compensate for that low-resource language. Additionally, introducing more language also reduces the variance. The proposed model significantly improves the PER to 55.0%.

Table.2 also demonstrates the performance of two neural models: the LSTM model and the transformer model. Interestingly, the neural model's performance does not perform better than the n-gram model when using a fixed language, even slightly worse than it. It is because the neural model further overfits the English dataset and could not capture orthography rules in other languages. The global model has the same trend, which again fails to fit each language. However, the nearest language model significantly reduces the error rate by almost 30%. Unlike the N-gram architecture, whose models of different languages are trained using a separate dataset, the neural model uses the shared architecture, and only distinguishes different languages by a language tag. This allows efficient parameter sharing between low-resource languages. Ensembling the model further reduces the error rate by more than 5%. In our experiment, the LSTM model and the transformer model have similar trends in their performance, but the LSTM model has a better performance than the transformer's one. The reason might be that there are far more hyperparameters to be tuned in the transformer model and the default sample configuration provided by the framework might not be optimal. As the main contribution of this work is to propose a general approach to ensemble languages rather than exploring different neural architectures, we only focus on how to ensemble models of different languages in this work.

4.4 Ensemble Analysis

It would be interesting to compare the number of languages when ensembling languages. Figure 4 demonstrates the influence of the number of languages from the LSTM model. PER drops quickly when we start ensembling models, it reaches the bottom when the number of nearest languages is 10, then starts to increase very slowly. We observe that there exists a bias-variance trade-off when changing the number of languages. When the number is relatively small, the prediction relies heavily on each language, therefore causing high variance when predicting for the target language. Increasing the number of languages could alleviate the variance problem, but using a large number of languages would decrease the accuracy as the selected languages are no longer close to the target language, which introduces more bias to the model.

To further understand the behavior of the model, we also show curves of Addition, Deletion, and Substitution in Figure.4. It indicates that after we start ensembling the model (from 2), the addition is increasing while the deletion is decreasing in general, the substitution decreases first and remains relatively flat later. The opposite trend of addition and deletion can be explained by the ensembling approach: when we introduce a new hypothesis into the model, it is probable some phonemes might not be aligned to the existing confusion set in the confusion network, to incorporate these new phonemes into the network, we need to create new confusion

Figure 4: The effect of using different number of nearest languages when ensembling models. It shows that we reach the best performance when we use the top-10 languages to ensemble outputs.

set, which would lead to more phoneme emissions. More phonemes would also contribute to decreasing the deletion rate as well. Therefore, that curve of PER is very similar to the curve of the substitution error (as the addition and deletion almost cancel each other). Not only does the ensemble model improve the substitution error quantitatively, it also improves the errors qualitatively: Table 3 shows the most frequent errors made by the nearest language model and the top-10 ensemble model. It indicates the most frequent substitution errors (/a/, /o/) and (/o/, /u/) are replaced by (/a/, /a:/) and (/i/, /i:/). We find latter errors are much closer to each other (they have phonological distances of 1, while the former errors have larger distances), therefore they are much better errors than the first two pairs qualitatively.

Errors	Most Common Errors
Add	/a/, /k/, /u/, /i/, /n/, /o/
Del	/a/, /i/, /ʔ/, /e/, /j/, /u/
Sub	(/a/, /o/), (/o/, /u/), (/r/, /l/), (/t/, /d/)
Add	/a/, /i/, /k/, /u/, /s/, /o/,
Del	/a/,/ʔ/, /i/, /e/, /u/ , /j/
Sub	(/r/, /l/),(/a:/, /a/), (/i:/, /i/), (/ɛ/, /e/)

Table 3: Most frequent errors in the LSTM model. The top half shows the errors in the nearest model, the bottom-half shows the errors when using 10 languages

5 Limitations

While we get reasonable performance in our testing languages, we acknowledge that there are several limitations in our approach: first, both of our training languages and testing languages are limited to languages available in Wiktionary. The full Glottolog Phylogenetic Tree has 110 top-level branches in total, however, our dataset only spans 40 branches. Therefore if we want to apply our approach to unseen languages in the remaining 70 branches, we have to depend on unrelated languages to build our ensemble model, which might lead to worse performance. Second, as our approach heavily depends on Glottolog, if the language is not available in the Glottolog database, then our approach cannot be applied to it. Finally, many of the 8k languages do not have orthographies, therefore it might be difficult or meaningless to evaluate the G2P performance for them.

6 Conclusion

In this work, we propose a zero-shot learning approach to build G2P models for 8k languages in the world. We use the phylogenetic tree to measure the distance between languages and combine multilingual outputs. We test our approach on 600 unseen languages and demonstrate it significantly outperforms baselines. We will release our datasets and models for 8k languages to allow more researchers to explore this direction.

References

Simon Ager. 2008. Omniglot-writing systems and languages of the world. *Retrieved January*, 27:2008.

Sercan Ö Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. 2017. Deep voice: Real-time neural text-to-speech. In *International Conference on Machine Learning*, pages 195–204. PMLR.

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

Alan W Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The third ESCA/COCOSDA workshop (ETRW) on speech synthesis*.

Harry Bleyan, Sandy Ritchie, Jonas Fromseier Mortensen, and Daan van Esch. 2019. Developing pronunciation models in new languages faster by exploiting common grapheme-to-phoneme correspondences across languages. In *INTERSPEECH*, pages 2100–2104.

CMU. 2000. The cmu pronunciation dictionary.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2009. *Introduction to algorithms*. MIT press.

Walter MP Daelemans and Antal PJ Van den Bosch. 1997. Language-independent data-oriented grapheme-to-phoneme conversion. In *Progress in speech synthesis*, pages 77–89. Springer.

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: Long Papers)*, pages 399–408.

Matthew S Dryer and Martin Haspelmath. 2013. The world atlas of language structures online.

Jonathan G Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 347–354. IEEE.

Ram Frost and Marian Katz. 1992. *Orthography, phonology, morphology and meaning*. Elsevier.

Kyle Gorman, Lucas FE Ashby, Aaron Goyzueta, Arya D McCarthy, Shijie Wu, and Daniel You. 2020. The sigmorphon 2020 shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 40–50.

Tomoki Hayashi, Ryuichi Yamamoto, Takenori Yoshimura, Peter Wu, Jiatong Shi, Takaaki Saeki, Yooncheol Ju, Yusuke Yasuda, Shinnosuke Takamichi, and Shinji Watanabe. 2021. Espnet2-tts: Extending the edge of tts research. *arXiv preprint arXiv:2110.07840*.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic inquiry*, 39(3):379–440.

Sittichai Jiampojamarn and Grzegorz Kondrak. 2010. Phoneme alignment: An exploration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 780–788.

Ronald M Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378.

Jackson L Lee, Lucas FE Ashby, M Elizabeth Garza, Yeonju Lee-Sikka, Sean Miller, Alan Wong, Arya D McCarthy, and Kyle Gorman. 2020. Massively multilingual pronunciation modeling with wikipron. In *Proceedings of the 12th language resources and evaluation conference*, pages 4223–4228.

Patrick Littell, David R Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. 2015. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE.

David R Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision g2p for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori S. Levin. 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. ACL.

Sebastian Nordhoff and Harald Hammarström. 2011. Glottolog/langdoc: Defining dialects, languages, and language families as collections of resources. In *First International Workshop on Linked Science 2011-In conjunction with the International Semantic Web Conference (ISWC 2011)*.

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.

Ben Peters, Jon Dehdari, and Josef van Genabith. 2017. Massively multilingual neural grapheme-to-phoneme conversion. *EMNLP 2017*, page 19.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 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. IEEE.

Eric Sven Ristad and Peter N Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Tim Schlippe, Wolf Quaschningk, and Tanja Schultz. 2014. Combining grapheme-to-phoneme converter outputs for enhanced pronunciation generation in low-resource scenarios. In *Spoken Language Technologies for Under-Resourced Languages*.

Terrence J Sejnowski and Charles R Rosenberg. 1987. Parallel networks that learn to pronounce english text. *Complex systems*, 1(1):145–168.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion.

Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth. 2019. Grapheme-to-phoneme conversion with convolutional neural networks. *Applied Sciences*, 9(6):1143.

Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth. 2020. Transformer based grapheme-to-phoneme conversion. *arXiv preprint arXiv:2004.06338*.