Linguistic Knowledge in Multilingual Grapheme-to-Phoneme Conversion

Roger Yu-Hsiang Lo
Department of Linguistics
The University of British Columbia
roger.lo@ubc.ca

Garrett Nicolai
Department of Linguistics
The University of British Columbia
garrett.nicolai@ubc.ca

Abstract

This paper documents the UBC Linguistics team's approach to the SIGMORPHON 2021 Grapheme-to-Phoneme Shared Task, concentrating on the low-resource setting. Our systems expand the baseline model with simple modifications informed by syllable structure and error analysis. In-depth investigation of test-set predictions shows that our best model rectifies a significant number of mistakes compared to the baseline prediction, besting all other submissions. Our results validate the view that careful error analysis in conjunction with linguistic knowledge can lead to more effective computational modeling.

1 Introduction

With speech technologies becoming ever more prevalent, grapheme-to-phoneme (G2P) conversion is an important part of the pipeline. G2P conversion refers to mapping a sequence of orthographic representations in some language to a sequence of phonetic symbols, often transcribed in the International Phonetic Alphabet (IPA). This is often an early step in tasks such as text-to-speech, where the pronunciation must be determined before any speech is produced. An example of such a G2P conversion, in Amharic, is illustrated below:

አማርኛ ↦ [amariɳːa] 'Amharic'

For the second year, one of SIGMORPHON shared tasks concentrates on G2P. This year, the task is further broken into three subtasks of varying data levels: high-resource (33K training instances), medium-resource (8K training instances), and low-resource (800 training instances). Our focus is on the low-resource subtask. The language data and associated constraints in the low-resource setting will be summarized in Section 3.1; the reader interested in the other two subtasks is referred to Ashby et al. (this volume) for an overview.

In this paper, we describe our methodology and approaches to the low-resource setting, including insights that informed our methods. We conclude with an extensive error analysis of the effectiveness of our approach.

This paper is structured as follows: Section 2 overviews previous work on G2P conversion. Section 3 gives a description of the data in the low-resource subtask, evaluation metric, and baseline results, along with the baseline model architecture. Section 4 introduces our approaches as well as the motivation behind them. We present our results in Section 5 and associated error analyses in Section 6. Finally, Section 7 concludes our paper.

2 Previous Work on G2P conversion

The techniques for performing G2P conversion have long been coupled with contemporary machine learning advances. Early paradigms utilize joint sequence models that rely on the alignment between grapheme and phoneme, usually with variants of the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). The resulting sequences of graphones (i.e., joint grapheme-phoneme tokens) are then modeled with n-gram models or Hidden Markov Models (e.g., Jiampojamarn et al., 2007; Bisani and Ney, 2008; Jiampojamarn and Kondrak, 2010). A variant of this paradigm includes weighted finite-state transducers trained on such graphone sequences (Novak et al., 2012, 2015).

With the rise of various neural network techniques, neural-based methods have dominated the scene ever since. For example, bidirectional long short-term memory-based (LSTM) networks using a connectionist temporal classification layer produce comparable results to earlier n-gram models (Rao et al., 2015). By incorporating alignment information into the model, the ceiling set by n-gram

Proceedings of the Seventeenth SIGMORPHON Workshop on Computational Research

in Phonetics, Phonology, and Morphology,pages 131–140

models has since been broken (Yao and Zweig, 2015). Attention further improved the performance, as attentional encoder-decoders (Toshniwal and Livescu, 2016) learned to focus on specific input sequences. As attention became "all that was needed" (Vaswani et al., 2017), transformer-based architectures have begun looming large (e.g., Yolchuyeva et al., 2019).

Recent years have also seen works that capitalize on multilingual data to train a single model with grapheme-phoneme pairs from multiple languages. For example, various systems from last year's shared task submissions learned from a multilingual signal (e.g., ElSaadany and Suter, 2020; Peters and Martins, 2020; Vesik et al., 2020).

3 The Low-resource Subtask

This section provides relevant information concerning the low-resource subtask.

3.1 Task Data

The provided data in the low-resource subtask come from ten languages[1]: Adyghe (`ady`; in the Cyrillic script), Modern Greek (`gre`; in the Greek alphabet), Icelandic (`ice`), Italian (`ita`), Khmer (`khm`; in the Khmer script, which is an alphasyllabary system), Latvian (`lat`), Maltese transliterated into the Latin script (`mlt_latn`), Romanian (`rum`), Slovene (`slv`), and the South Wales dialect of Welsh (`wel_sw`). The data are extracted from Wikitionary[2] using WikiPron (Lee et al., 2020), and filtered and downsampled with proprietary techniques, resulting in each language having 1,000 labeled grapheme-phoneme pairs, split into a training set of 800 pairs, a development set of 100 pairs, and a blind test set of 100 pairs.

3.2 The Evaluation Metric

This year, the evaluation metric is the word error rate (WER), which is simply the percentage of words for which the predicted transcription sequence differs from the ground-truth transcription. Different systems are ranked based on the macro-average over all languages, with lower scores indicating better systems. We also adopted this metric when evaluating our models on the development sets.

[1] All output is represented in IPA; unless specified otherwise, the input is written in the Latin alphabet.

[2] `https://www.wiktionary.org/`

3.3 Baselines

The official baselines for individual languages are based on an ensembled neural transducer trained with the imitation learning (IL) paradigm (Makarov and Clematide, 2018a). The baseline WERs are tabulated in Table 3. In what follows, we overview this baseline neural-transducer system, as our models are built on top of this system. The detailed formal description of the baseline system can be found in Makarov and Clematide (2018a,b,c, 2020).

The neural transducer in question defines a conditional distribution over edit actions, such as copy, deletion, insertion, and substitution:

$$p_\theta(\boldsymbol{y}, \boldsymbol{a}|x) = \prod_{j=1}^{|\boldsymbol{a}|} p_\theta(a_j|a_{<j}, \boldsymbol{x}),$$

where $\boldsymbol{x}$ denotes an input sequence of graphemes, and $\boldsymbol{a} = a_1 \dots a_{|\boldsymbol{a}|}$ stands for a sequence of edit actions. Note that the ouput sequence $\boldsymbol{y}$ is missing from the conditional probability on the right-hand side as it can be deterministically computed from $\boldsymbol{x}$ and $\boldsymbol{a}$. The model is implemented with an LSTM decoder, coupled with a bidirectional LSTM encoder.

The model is trained with IL and therefore demands an expert policy, which contains demonstrations of how the task can be optimally solved given any configuration. Cast as IL, the mapping from graphemes to phonemes can be understood as following an optimal path dictated by the expert policy that gradually turns input orthographic symbols to output IPA characters. To acquire the expert policy, a Stochastic Edit Distance (Ristad and Yianilos, 1998) model trained with the EM algorithm is employed to find an edit sequence consisting of four types of edits: copy, deletion, insertion, and substitution. During training time, the expert policy is queried to identify the next optimal edit that minimizes the following objective expressed in terms of Levenshtein distance and edit sequence cost:

$$\beta \text{ED}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \text{ED}(\boldsymbol{x}, \hat{\boldsymbol{y}}), \beta \geq 1,$$

where the first term is the Levenshtein distance between the target sequence $\boldsymbol{y}$ and the predicted sequence $\hat{\boldsymbol{y}}$, and the second term measures the cost of editing $\boldsymbol{x}$ to $\hat{\boldsymbol{y}}$.

The baseline is run with default hyperparameter values, which include ten different initial seeds and a beam of size 4 during inference. The predictions of these individual models are ensembled using a

voting majority. Early efforts to modify the ensemble to incorporate system confidence showed that a majority ensemble was sufficient.

This model has proved to be competitive, judging from its performance on the previous year's G2P shared task. We therefore decided to use it as the foundation to construct our systems.

4 Our Approaches

This section lays out our attempted approaches. We investigate two alternatives, both linguistic in nature. The first is inspired by a universal linguistic structure—the syllable—and the other by the error patterns discerned from the baseline predictions on the development data.

4.1 System 1: Augmenting Data with Unsupervised Syllable Boundaries

Our first approach originates from the observation that, in natural languages, a sequence of sounds does not just assume a flat structure. Neighboring sounds group to form units, such as the onset, nucleus, and coda. In turn, these units can further project to a syllable (see Figure 1 for an example of such projection). Syllables are useful structural units in describing various linguistic phenomena and indeed in predicting the pronunciation of a word in some languages (e.g., Treiman, 1994). For instance, in Dutch, the vowel quality of the nucleus can be reliably inferred from the spelling after proper syllabification: *.dag.* [d**ɑ**x] 'day' but *.da.gen.* [d**aː**ɣən] 'days', where . marks syllable boundaries. Note that *a* in a closed syllable is pronounced as the short vowel [ɑ], but as the long vowel [aː] in an open syllable. In applying syllabification to G2P conversion, van Esch et al. (2016) find that training RNNs to jointly predict phoneme sequences, syllabification, and stress leads to further performance gains in some languages, compared to models trained without syllabification and stress information.

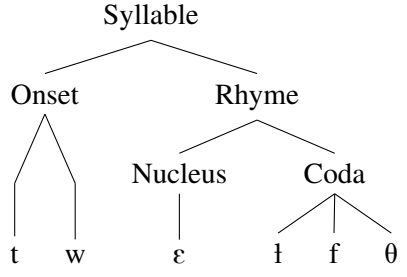

Figure 1: The syllable structure of *twelfth* [twɛɬfθ]

To identify syllable boundaries in the input sequence, we adopted a simple heuristic, the specific steps of which are listed below:[3]

1. **Find vowels in the output:** We first identify the vowels in the phoneme sequence by comparing each segment with the vowel symbols from the IPA chart. For instance, the symbols [ø] and [y] in [tʰrøyst] for Icelandic *traust* are vowels because they match the vowel symbols [ø] and [y] on the IPA chart.

2. **Find vowels in the input:** Next we align the grapheme sequence with the phoneme sequence using an unsupervised many-to-many aligner (Jiampojamarn et al., 2007; Jiampojamarn and Kondrak, 2010). By identifying graphemes that are aligned to phonemic vowels, we can identify vowels in the input. Using the Icelandic example again, the aligner produces a one-to-one mapping: *t* ↦ tʰ, *r* ↦ r, *a* ↦ ∅, *u* ↦ y, *s* ↦ s, and *t* ↦ t. We therefore assume that the input characters *a* and *u* represent two vowels. Note that this step is often redundant for input sequences based on the Latin script but is useful in identifying vowel symbols in other scripts.

3. **Find valid onsets and codas:** A key step in syllabification is to identify which sequences of consonants can form an onset or a coda. Without resorting to linguistic knowledge, one way to identify valid onsets and codas is to look at the two ends of a word—consonant sequences appearing word-initially before the first vowel are valid onsets, and consonant sequences after the final vowel are valid codas. Looping through each input sequence in the training data gives us a list of valid onsets and codas. In the Icelandic example *traust*, the initial *tr* sequence must be a valid onset, and the final *st* sequence a valid coda.

4. **Break word-medial consonant sequences into an onset and a coda:** Unfortunately identifying onsets and codas among word-medial consonant sequences is not as straightforward. For example, how do we know the

[3] We are aware that different languages permit distinct syllable constituents (e.g., some languages allow syllabic consonants while others do not), but given the restriction that we are not allowed to use external resources in the low-resource subtask, we simply assume that all syllables must contain a vowel.

sequence in the input V*ngstr*V (V for a vowel character) should be parsed as V*ng.str*V, as V*n.gstr*V, or even as V.*ngstr*V? To tackle this problem, we use the valid onset and coda lists gathered from the previous step: we split the consonant sequence into two parts, and we choose the split where the first part is a valid coda and the second part a valid onset. For instance, suppose we have an onset list {*str*, *tr*} and a coda list {*ng*, *st*}. This implies that we only have a single valid split—V*ng.str*V—so *ng* is treated as the coda for the previous syllable and *str* as the onset for the following syllable. In the case where more than one split is acceptable, we favor the split that produces a more complex onset, based on the linguistic heuristic that natural languages tend to tolerate more complex onsets than codas. For example, V*ng.str*V > V*ngs.tr*V. In the situation where none of the splits produces a concatenation of a valid coda and onset, we adopt the following heuristic:

- If there is only one medial consonant (such as in the case where the consonant can only occur word-internally but not in the onset or coda position), this consonant is classified as the onset for the following syllable.
- If there is more than one consonant, the first consonant is classified as the coda and attached to the previous syllable while the rest as the onset of the following syllable.

Of course, this procedure is not free of errors (e.g., some languages have onsets that are only allowed word-medially, so word-initial onsets will naturally not include them), but overall it gives reasonable results.

5. **Form syllables:** The last step is to put together consonant and vowel characters to form syllables. The simplest approach is to allow each vowel character to be projected as a nucleus and distribute onsets and codas around these nuclei to build syllables. If there are four vowels in the input, there are likewise four syllables. There is one important caveat, however. When there are two or more consecutive vowel characters, some languages prefer to merge them into a single vowel/nucleus in their pronunciation (e.g., Greek χα͡ι ↦ [c**e**]) while other languages simply default to vowel hiatuses/two side-by-side nuclei (e.g., Italian *badi͡a* ↦ [bad**ia**])—indeed, both are common cross-linguistically. We again rely on the alignment results in the second step to select the vowel segmentation strategy for individual languages.

After we have identified the syllables that compose each word, we augmented the input sequences with syllable boundaries. We identify four labels to distinguish different types of syllable boundaries: `<cc>`, `<cv>`. `<vc>`, and `<vv>`, depending on the classes of sound the segments straddling the syllable boundary belong to. For instance, the input sequence `b í l a v e r k s t æ ð i` in Icelandic will be augmented to be `b í <vc> l a <vc> v e r k <cc> s t æ <vc> ð i`. We applied the same syllabification algorithm to all languages to generate new input sequences, with the exception of Khmer, as the Khmer script does not permit a straightforward linear mapping between input and output sequences, which is crucial for the vowel identification step. We then used these syllabified input sequences, along with their target transcriptions, as the training data for the baseline model.[4]

4.2 System 2: Penalizing Vowel and Diacritic Errors

Our second approach focuses on the training objective of the baseline model, and is driven by the errors we observed in the baseline predictions. Specifically, we noticed that the majority of errors for the languages with a high WER—Khmer, Latvian, and Slovene—concerned vowels, some examples of which are given in Table 1. Note the nature of these mistakes: the mismatch can be in the vowel quality (e.g., [ɔ] for [o]), in the vowel length (e.g., [áː] for [á]), in the pitch accent (e.g., [íː] for [ìː]), or a combination thereof.

Based on the above observation, we modified the baseline model to explicitly address this vowel-mismatching issue. We modified the objective such that erroneous vowel or diacritic (e.g., the lengthening marker [ː]) predictions during training incur

[4]The hyperparameters used are the default values provided in the baseline model code: character and action embedding = 100, encoder LSTM state dimension = decoder LSTM state dimension = 200, encoder layer = decoder layer = 1, beam width = 4, roll-in hyperparameter = 1, epochs = 60, patience = 12, batch size = 5, EM iterations = 10, ensemble size = 10.

Language	Target	Baseline prediction
khm	n **u** h	n **ŭ ə** h
	r **ɔː** j	r **ĕ ə** j
	s p **ŏ ə** n	s p **a** n
lat	t s **eː** l s	t s **êː** l s
	j **u ō** k s	j **ù o** k s
	v **æ̂** l s	v **ǣː** l s
slv	j **óː** g **u** r t	j ɔ g **úː** r t
	k r **ìː** ʃ	k r **íː** ʃ
	z d **á** j	z d **áː** j

Table 1: Typical errors in the development set that involve vowels from Khmer (khm), Latvian (lat), and Slovene (slv)

additional penalties. Each incorrectly-predicted vowel incurs this penalty. The penalty acts as a regularizer that forces the model to expend more effort on learning vowels. This modification is in the same spirit as the softmax-margin objective of Gimpel and Smith (2010), which penalizes high-cost outputs more heavily, but our approach is even simpler—we merely supplement the loss with additional penalties for vowels and diacritics. We fine-tuned the vowel and diacritic penalties using a grid search on the development data, incrementing each by 0.1, from 0 to 0.5. In the cases of ties, we skewed higher as the penalties generally worked better at higher values. The final values used to generate predictions for the test data are listed in Table 2. We also note that the vowel penalty had significantly more impact than the diacritic penalty.

	Penalty	
Language	Vowel	Diacritic
ady	0.5	0.3
gre	0.3	0.2
ice	0.3	0.3
ita	0.5	0.5
khm	0.2	0.4
lav	0.5	0.5
mlt_latn	0.2	0.2
rum	0.5	0.2
slv	0.4	0.4
wel_sw	0.4	0.5

Table 2: Vowel penalty and diacritic penalty values in the final models

5 Results

The performances of our systems, measured in WER, are juxtaposed with the official baseline results in Table 3. We first note that the baseline was particularly strong—gains were difficult to achieve for most languages. Our first system (Syl), which is based on syllabic information, unfortunately does not outperform the baseline. Our second system (VP), which includes additional penalties for vowels and diacritics, however, does outperform the baselines in several languages. Furthermore, the macro WER average not only outperforms the baseline, but all other submitted systems.

	WER		
Language	Baseline	Syl	VP
ady	22	25	22
gre	21	22	22
ice	12	13	**11**
ita	19	20	22
khm	34	**31**	**28**
lav	55	58	**49**
mlt_latn	19	19	**18**
rum	10	14	10
slv	49	56	**47**
wel_sw	10	13	12
Average	25.1	27.1	**24.1**

Table 3: Comparison of test-set results based on the word error rates (WERs)

It seems that extra syllable information does not help with predictions in this particular setting. It might be the case that additional syllable boundaries increase input variability without providing much useful information with the current neural-transducer architecture. Alternatively, information about syllable boundary locations might be redundant for this set of languages. Finally, it is possible that the unsupervised nature of our syllable annotation was too noisy to aid the model. We leave these speculations as research questions for future endeavors and restrict the subsequent error analyses and discussion to the results from our vowel-penalty system.[5]

[5]One reviewer raised a question of why only syllable boundaries, as opposed to smaller constituents, such as onsets or codas, are marked. Our hunch is that many phonological alternations happen at syllable boundaries, and that vowel length in some languages depends on whether the nucleus vowel is in a closed or open syllable. Also, given that adding syllable

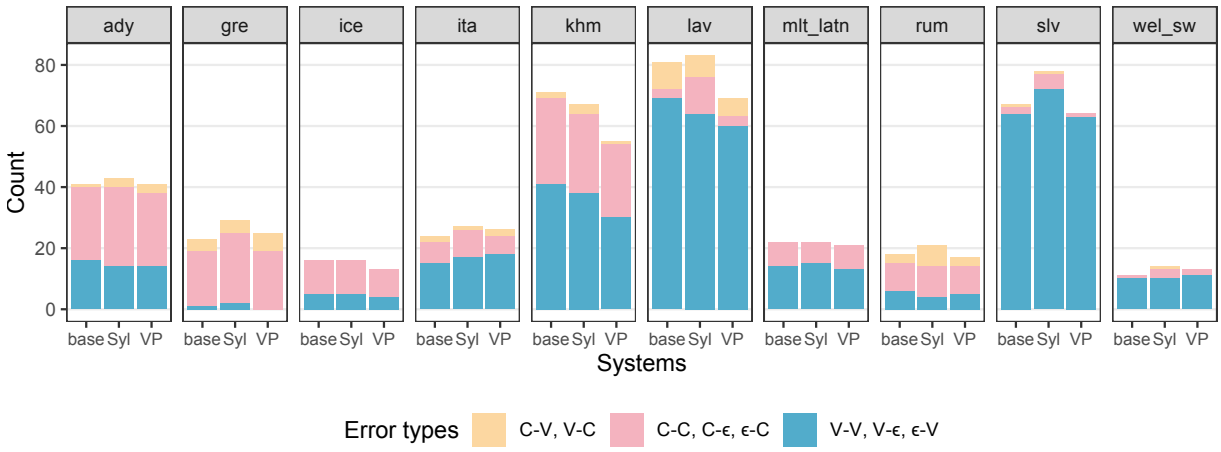

Figure 2: Distributions of error types in test-set predictions across languages. Error types are distinguished based on whether an error involves only consonants, only vowels, or both. For example, C-V means that the error is caused by a ground-truth consonant being replaced by a vowel in the prediction. C-ϵ means that it is a deletion error where the ground-truth consonant is missing in the prediction while ϵ-C represents an insertion error where a consonant is wrongly added.

6 Error Analyses

In this section, we provide detailed error analyses on the test-set predictions from our best system. The goals of these analyses are twofold: (i) to examine the aspects in which this model outperforms the baseline and to what extent, and (ii) to get a better understanding of the nature of errors made by the system—we believe that insights and improvements can be derived from a good grasp of error patterns.

We analyzed the mismatches between predicted sequences and ground-truth sequences at the segmental level. For this purpose, we again utilized many-to-many alignment (Jiampojamarn et al., 2007; Jiampojamarn and Kondrak, 2010), but this time between a predicted sequence and the corresponding ground-truth sequence.[6] For each error along the aligned sequence, we classified it into one of the three kinds:

- Those involving erroneous vowel insertions (e.g., $\epsilon \rightarrow$ [ə]), deletions (e.g., [ə] $\rightarrow \epsilon$), or substitutions (e.g., [ə] $\rightarrow$ [a]).
- In the same vein, those involving erroneous consonant insertions (e.g., $\epsilon \rightarrow$ [ʔ]), deletions (e.g., [ʔ] $\rightarrow \epsilon$), and substitutions (e.g., [d] $\rightarrow$ [t]).
- Those involving exchanges of a vowel and a consonant (e.g., [w] $\rightarrow$ [u]) or vice versa.

The frequency of each error type made by the baseline model and our systems for each individual language is plotted in Figure 2. Some patterns are immediately clear. First, both systems have a similar pattern in terms of the distribution of error types across language, albeit that ours makes fewer errors on average. Second, both systems err on different elements, depending on the language. For instance, while Adyghe (`ady`) and Khmer (`khm`) have a more balanced distribution between consonant and vowel errors, Slovene (`slv`) and Welsh (`wel_sw`) are dominated by vowel errors. Third, the improvements gained in our system seem to come mostly from reduction in vowel errors, as is evident in the case of Khmer, Latvian (`lav`), and, to a lesser extent, Slovene.

The final observation is backed up if we zoom in on the errors in these three languages, which we visualize in Figure 3. Many incorrect vowels generated by the baseline model are now correctly predicted. We note that there are also cases, though less common, where the baseline model gives the right prediction, but ours does not. It should be pointed out that, although our system shows improvement over the baseline, there is still plenty of room for improvement in many languages, and our system still produces incorrect vowels in many

boundaries does not improve the results, it is unlikely that marking constituent boundaries, which adds more variability to the input, will result in better performance, though we did not test this hypothesis.

[6]The parameters used are: allowing deletion of input grapheme strings, maximum length of aligned grapheme and phoneme substring being one, and a training threshold of 0.001.

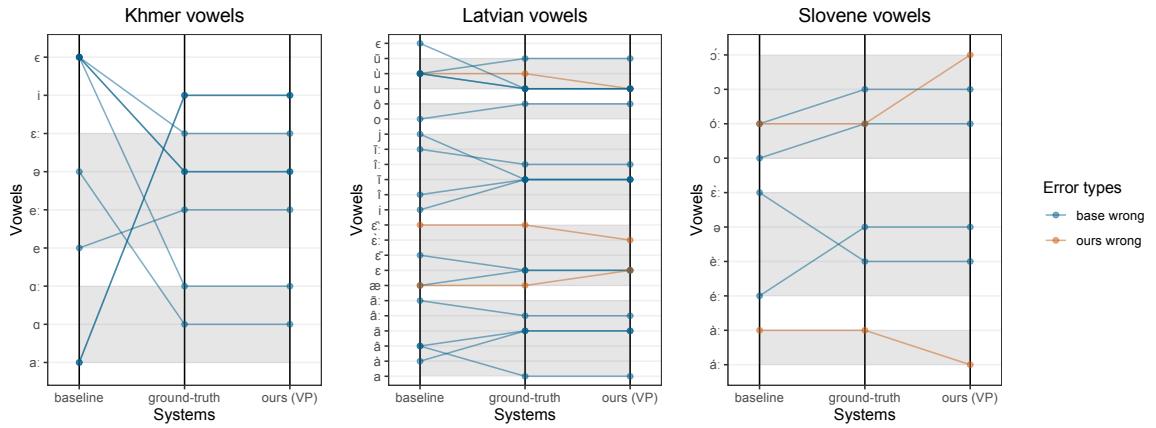

Figure 3: Comparison of vowels predicted by the baseline model and our best system (VP) with the ground-truth vowels. Here we only visualize the cases where either the baseline model gives the right vowel but our system does not, or vice versa. We do not include cases where both the baseline model and our system predict the correct vowel, or both predict an incorrect vowel, to avoid cluttering the view. Each baseline—ground-truth—ours line represents a set of aligned vowels in the same word; the horizontal line segment between a system and the ground-truth means that the prediction from the system agrees with the ground-truth. Color hues are used to distinguish cases where the prediction from the baseline is correct versus those where the prediction from our second system is correct. Shaded areas on the plots enclose vowels of similar vowel quality.

instances.

Finally, we look at several languages which still resulted in high WER on the test set—`ady`, `gre`, `ita`, `khm`, `lav`, and `slv`. We analyze the confusion matrix analysis to identify clusters of commonly-confused phonemes. This analysis again relies on the alignment between the ground-truth sequence and the corresponding predicted sequence to characterize error distributions. The results from this analysis are shown in Figure 4, and some interesting patterns are discussed below. Figure 2 suggests that Khmer has an equal share of consonant and vowel errors, and the heat maps in Figure 4 reveal that these errors do not seem to follow a certain pattern. However, a different picture emerges with Latvian and Slovene. For both languages, Figure 2 indicates the dominance of errors tied to vowels; consonant errors account for a relatively small proportion of errors. This observation is borne out in Figure 4, with the consonant heat maps for the two languages displaying a clear diagonal stripe, and the vowel heat maps showing much more off-diagonal signals. What is more interesting is that the vowel errors in fact form clusters, as highlighted by white squares on the heat maps. The general pattern is that confusion only arises *within* a cluster where vowels are of similar quality but differ in terms of length or pitch accent. For example, while [iː] might be incorrectly-predicted as [i], our model does not confuse it with, say, [u]. The challenges these languages present to the models are therefore largely suprasegmental—vowel length and pitch accent, both of which are lexicalized and not explicitly marked in the orthography. For the other three languages, their errors also show distinct patterns: for Adyghe, consonants differing only in secondary features can get confused; in Greek, many errors can be attributed to the mixing of [r] and [ɾ]; in Italian, front and back mid vowels can trick our model.

We hope that our detailed error analyses show not only that these errors "make linguistic sense"—and therefore attest to the power of the model—but also point out a pathway along which future modeling can be improved.

7 Conclusion

This paper presented the approaches adopted by the UBC Linguistics team to tackle the SIGMORPHON 2021 Grapheme-to-Phoneme Conversion challenge in the low-resource setting. Our submissions build upon the baseline model with modifications inspired by syllable structure and vowel error patterns. While the first modification does not result in more accurate predictions, the second modification does lead to sizable improvements over the baseline results. Subsequent error analyses reveal that the modified model indeed reduces erroneous vowel predictions for languages whose errors are dominated by vowel mismatches. Our approaches also demonstrate that patterns uncov-

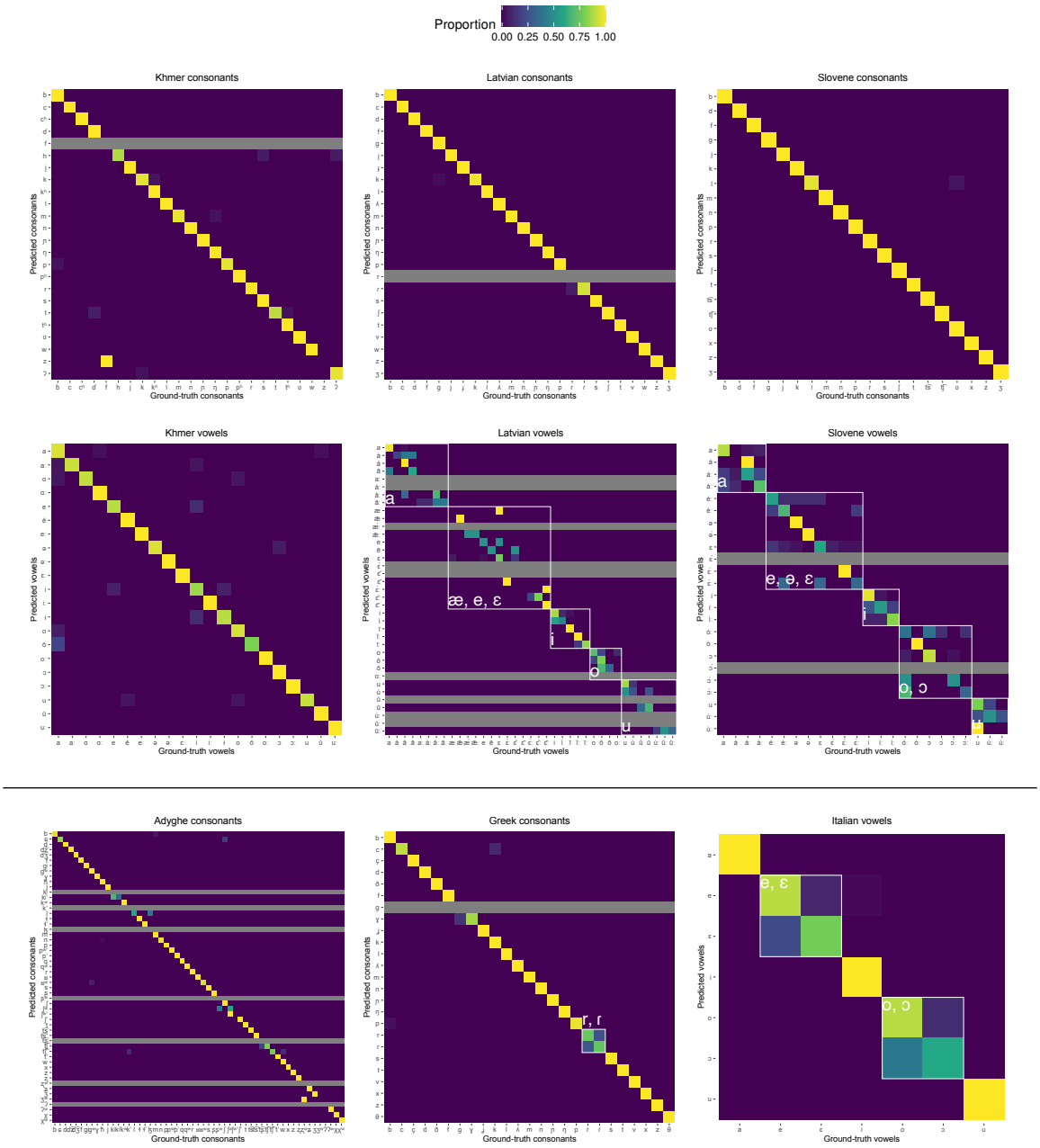

Figure 4: Confusion matrices of vowel and consonant predictions by our second system (VP) for languages with the test WER $> 20\%$. Each row represents a predicted segment, with colors across columns indicating the proportion of times the predicted segment matches individual ground-truth segments. A gray row means the segment in question is absent in any predicted phoneme sequences but is present in at least one ground-truth sequence. The diagonal elements represent the number of times for which the predicted segment matches the target segment, while off-diagonal elements are those that are mis-predicted by the system. White squares are added to highlight segment groups where mismatches are common.

ered from careful error analyses can inform the directions for potential improvements.

References

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

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Omnia ElSaadany and Benjamin Suter. 2020. Grapheme-to-phoneme conversion with a multilingual transformer model. In *Proceedings of the Seventeenth SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 85–89.

Daan van Esch, Mason Chua, and Kanishka Rao. 2016. Predicting pronunciations with syllabification and stress with recurrent neural networks. In *Proceedings of Interspeech 2016*, pages 2841–2845.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, pages 733–736.

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

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and Hidden Markov Models to letter-to-phoneme conversion. In *Proceedings of NAACL HLT 2007*, pages 372–379.

Jackson L. Lee, Lucas F. E. Ashby, M. Elizabeth Garza, Yeonju Lee-Sikka, Sean Miller, Alan Wong, Arya D. McCarthy, and Kyle Gorman. 2020. Massively multilingual pronunciation mining with WikiPron. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 4223–4228.

Peter Makarov and Simon Clematide. 2018a. Imitation learning for neural morphological string transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2877–2882.

Peter Makarov and Simon Clematide. 2018b. Neural transition-based string transduction for limited-resource setting in morphology. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 83–93.

Peter Makarov and Simon Clematide. 2018c. UZH at CoNLL-SIGMORPHON 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL-SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 69–75.

Peter Makarov and Simon Clematide. 2020. CLUZH at SIGMORPHON 2020 shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the Seventeenth SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 171–176.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 45–49.

Josef Robert Novak, Nobuaki Minematsu, and Keikichi Hirose. 2015. Phonetisaurus: Exploring garpheme-to-phoneme conversion with joint n-gram models in the WFST framework. *Natural Language Engineering*, 22(6):907–938.

Ben Peters and André F. T. Martins. 2020. DeepSPIN at SIGMORPHON 2020: One-size-fits-all multilingual models. In *Proceedings of the Seventeenth SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 63–69.

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

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

Shubham Toshniwal and Karen Livescu. 2016. Jointly learning to align and convert graphemes to phonemes with neural attention models. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 76–82.

Rebecca Treiman. 1994. To what extent do orthographic units in print mirror phonological units in speech? *Journal of Psycholinguistic Research*, 23(1):91–110.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukaaz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, pages 1–11.

Kaili Vesik, Muhammad Abdul-Mageed, and Miikka Silfverberg. 2020. One model to pronounce them all: Multilingual grapheme-to-phoneme conversion with a Transformer ensemble. In *Proceedings of the*

Seventeenth SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 146–152.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *Proceedings of Interspeech 2015*, pages 3330–3334.

Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth. 2019. Transformer based grapheme-to-phoneme conversion. In *Proceedings of Interspeech 2019*, pages 2095–2099.