Frustratingly Easy Multilingual Grapheme-to-Phoneme Conversion

Nikhil Prabhu and **Katharina Kann**
University of Colorado Boulder
{nikhil.prabhu,katharina.kann}@colorado.edu

Abstract

In this paper, we describe two CU Boulder submissions to SIGMORPHON 2020 Task 1 on multilingual grapheme-to-phoneme conversion (G2P). Inspired by the high performance of a standard transformer model (Vaswani et al., 2017) on the task, we improve over this approach by adding two modifications: (i) Instead of training exclusively on G2P, we additionally create examples for the opposite direction, phoneme-to-grapheme conversion (P2G). We then perform multi-task training on both tasks. (ii) We produce ensembles of our models via majority voting. Our approaches, though being conceptually simple, result in systems that place 6th and 8th amongst 23 submitted systems, and obtain the best results out of all systems on Lithuanian and Modern Greek, respectively.

1 Introduction

This paper describes the CU Boulder submissions to the SIGMORPHON 2020 shared task on multilingual grapheme-to-phoneme conversion (G2P). G2P is an important task, due to its applications in text-to-speech and automatic speech recognition systems. It is explained by Jurafsky and Martin (2009) as:

> The process of converting a sequence of letters into a sequence of phones is called grapheme-to-phoneme conversion, sometimes shortened g2p. The job of a grapheme-to-phoneme algorithm is thus to convert a letter string like *cake* into a phone string like [K EY K].

While the earliest G2P algorithms have used handwritten parser-based rules in the format of Chomsky-Halle rewrite rules, often called letter-to-sound, or LTS, rules (Chomsky and Halle, 1968), later techniques have moved on to generating semi-automatic alignment tables such as in Pagel et al. (1998). Today, a lot of work aims at using machine learning – in particular deep learning techniques – to solve sequence-to-sequence problems like this.

We explore using a transformer model (Vaswani et al., 2017) for this problem, since it has shown great promise in several areas of natural language processing (NLP), outperforming the previous state of the art on a large variety of tasks, including machine translation (Vaswani et al., 2017), summarization (Raffel et al., 2019), question-answering (Raffel et al., 2019), and sentiment-analysis (Munikar et al., 2019). While previous work has used transformers for G2P, experiments have only been performed on English, specifically on the CMUDict (Weide, 2005) and NetTalk[1] datasets (Yolchuyeva et al., 2020; Sun et al., 2019). Our approach builds upon the standard architecture by adding two straightforward modifications: multi-task training (Caruana, 1997) and ensembling. We find that these simple additions lead to performance improvements over the standard model, and our models place 6th and 8th among 23 submissions to the SIGMORPHON 2020 shared task on multilingual grapheme-to-phoneme conversion. Our two models further perform the best on the languages Lithuanian and Modern Greek, respectively.

2 Task and Background

2.1 Grapheme-to-Phoneme Conversion

G2P can be cast as a sequence-to-sequence task, where the input sequence is a sequence of graphemes, i.e., the spelling of a word, and the output sequence is a sequence of IPA-like symbols, representing the pronunciation of the same word.

[1] https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Nettalk+Corpus)

Formally, let Σ_G be an alphabet of graphemes and Σ_P be an alphabet of phonemes. For a word w in a language, G2P then refers to the mapping

$$g(w) \mapsto p(w), \tag{1}$$

with $g(w) \in \Sigma_G^*$ and $p(w) \in \Sigma_P^*$ being the grapheme and phoneme representations of w, respectively.

2.2 Related Work

Many different approaches to G2P exist in the literature, including rule-based systems (Black et al., 1998), LSTMs (Rao et al., 2015), joint-sequence models (Galescu and Allen, 2002), and encoder-decoder architectures, based on convolutional neural networks (Yolchuyeva et al., 2019), LSTMs (Yao and Zweig, 2015), or transformers (Yolchuyeva et al., 2020; Sun et al., 2019). In this paper, we improve over previous work by exploring two straightforward extensions of a standard transformer (Vaswani et al., 2017) model for the task: multi-task training (Caruana, 1997) and ensembling. Multi-task training has been explored previously for G2P (Milde et al., 2017), with the tasks being training on different languages and alphabet sets. Sun et al. (2019) successfully used token-level ensemble distillation for G2P to boost accuracy and reduce model-size, ensembling models based on multiple different architectures.

3 Proposed Approach

We submit two different systems to the shared task, which are based on the transformer architecture, multi-task learning, and ensembling. We describe all components individually in this section.

3.1 Model

Our model architecture is shown in Figure 1; the vanilla transformer proposed by Vaswani et al. (2017). In short, the transformer is an auto-regressive encoder-decoder architecture, which uses stacked self-attention and fully-connected layers for both the encoder and decoder. The decoder is connected to the encoder via multi-head attention over the encoder outputs. Details can be found in the original paper.

3.2 Multi-task Training

We propose to train our model jointly on two tasks: (i) G2P and (ii) phoneme-to-grapheme conversion

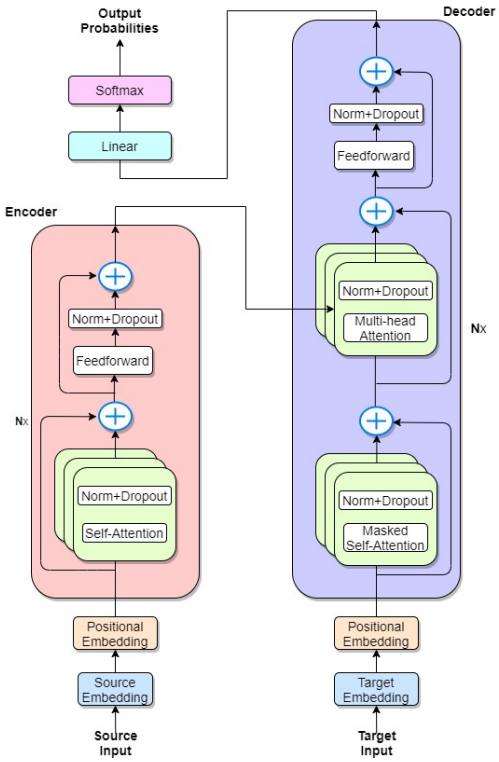

Figure 1: The transformer model architecture.

Hyperparameter	Value
Batch Size	128
Embedding Dimension	256
Hidden Dimension	1024
Dropout	0.3
Number of Encoder Layers	4
Number of Decoder Layers	4
Number of Attention Heads	4
Learning Rate	1e-3
β_1	0.9
β_2	0.998
Label Smoothing Coefficient	0.1
Max Norm (Gradient clipping)	1

Table 1: The hyperparameters used in our experiments.

(P2G). Using our formalization from before, given a word w, P2G corresponds to the mapping

$$p(w) \mapsto g(w). \tag{2}$$

We denote the set of our original G2P training examples as D_{g2p} and our P2G examples, which we obtain by inverting all examples in D_{g2p}, as D_{p2g}. We then aim to obtain model parameters θ that maximize the joint log-likelihood of both datasets:

$$\mathcal{L}(\theta) = \sum_{(w \in D_{g2p})} \log p_\theta(p[w] \mid g[w], \lambda_g) + \sum_{(w \in D_{p2g})} \log p_\theta(g[w] \mid p[w], \lambda_p) \tag{3}$$

Grapheme	Phoneme	Phoneme	Grapheme
! a a n d a c h t	a: n d ɑ x t	? a: n d ɑ x t	a a n d a c h t
! b a s s o n	b ɑ s ɔ n	? b ɑ s ɔ n	b a s s o n
! b e g i n t	b ə ɣ ɪ n t	? b ə ɣ ɪ n t	b e g i n t
! g i e r s t	x i: r s t	? x i: r s t	g i e r s t
! h e u p	ɦ ø: p	? ɦ ø: p	h e u p

Table 2: G2P (left) and P2G (right).

$\lambda_g, \lambda_p \notin \Sigma_G \cup \Sigma_P$ are special symbols which we prepend to each input. These so-called *task-embeddings* indicate to our model which task each individual input belongs to. Examples for both tasks are shown in Table 2.

Intuition. By training our model jointly on G2P and P2G, we expect it to learn properties that both tasks have in common. First, both tasks require learning of a monotonic left-to-right mapping. Second, for some languages, $\Sigma_G \cap \Sigma_P \neq \emptyset$, cf. Table 2 for Dutch as an example. Symbols in $\Sigma_G \cap \Sigma_P$ are commonly mapped onto each other in both directions, such that we expect the model to learn this from both tasks.

3.3 Ensembling

Our second straightforward modification of the standard transformer model is that we create ensembles via majority voting. In particular, each of our two submitted systems is an ensemble of multiple different models for each language, which we generate using different random seeds. We then create predictions with all models participating in each ensemble, and choose the solution that occurs most frequently, with ties being broken randomly.

Our first submitted model – **CU-1** – is an ensemble of 5 standard G2P transformers and 5 multi-task transformers. Our second system – **CU-2** – is an ensemble of 5 multi-task transformers.

4 Experiments

4.1 Data

The datasets provided for the shared task spans 15 individual languages, with each training set consisting of 3600 pairs of graphemes and their associated phonemes. The datasets include an initial set of core languages – Armenian (arm), Bulgarian (bul), French (fre), Georgian (geo), Modern Greek (gre), Hindi (hin), Hungarian (hun), Icelandic (ice), Korean (kor), and Lithaunian (lit) –, and a set of surprise languages, which have been released shortly before the shared task deadline – Adyghe (ady), Dutch (dut), Japanese (hiragana) (jap), Romanian (rum), and Vietnamese (vie). The data is primarily extracted from Wiktionary using the wikipron library (Lee et al., 2020).

	CU-1		CU-2		CU-TB	
	WER	PER	WER	PER	WER	PER
arm	13.56	2.75	14.22	2.94	16.10	3.37
bul	29.11	6.98	30.22	7.41	32.06	7.32
fre	8.00	2.00	8.00	1.84	10.29	2.65
geo	25.33	4.98	24.67	4.83	26.03	5.20
gre	17.56	3.05	17.78	3.14	17.92	3.40
hin	6.22	1.58	6.89	1.78	8.78	2.28
hun	2.89	0.66	3.11	0.60	4.52	1.03
ice	9.78	2.13	9.11	2.13	12.20	2.83
kor	23.11	6.83	24.22	6.61	26.61	7.43
lit	21.56	4.11	22.44	4.18	22.88	4.41
ady	22.89	5.68	23.11	5.68	24.66	6.33
dut	14.67	2.84	14.44	2.63	16.80	3.46
jap	6.67	2.14	6.67	2.18	7.23	2.42
rum	12.22	3.15	12.00	3.09	13.04	3.37
vie	2.89	1.07	2.89	0.99	4.70	1.60
avg.	**14.43**	**3.33**	**14.65**	**3.34**	**16.25**	**3.80**

Table 3: Development results in WER and PER; CU-1=standard and multi-task transformer ensemble, CU-2=multi-task transformer ensemble, CU-TB=standard transformer ensemble.

4.2 Hyperparameters

Following the official shared task baseline, we employ the hyperparameters shown in Table 1. All models are trained for 150 epochs. Starting from epoch 100, we evaluate every 5 epochs for early stopping. Encoder and decoder embeddings are tied, and the maximum sequence length is 24. Our system is built on the transformer implementation by Wu et al. (2020), and our final code is available on github.[2]

4.3 Metrics

Word error rate (WER). Word error rate is the percentage of words for which the model's prediction does not exactly match the gold transcription.
Phoneme error rate (PER). Phoneme error rate is the percentage of wrong characters in the model's prediction as compared to the gold standard.

Both metrics are calculated using the official evaluation script[3] provided for the shared task.

4.4 Development Results

The results on the development sets are shown in Table 3. CU-TB represents a transformer baseline

[2] https://github.com/NikhilPr95/neural-transducer

[3] https://github.com/sigmorphon/2020/blob/master/task1/evaluation/evaluate.py

	CU-1		CU-2		SIG-TB		SIG-LSTM	
	WER	PER	WER	PER	WER	PER	WER	PER
arm	12.89	2.91	13.56	3.04	14.22	3.29	14.67	3.49
bul	26.89	5.65	29.78	6.30	34.00	7.89	31.11	5.94
fre	5.78	1.48	5.56	1.28	6.89	1.72	6.22	1.32
geo	25.78	4.83	27.11	5.08	28.00	5.43	26.44	5.14
gre	15.11	2.51	14.44	2.42	18.89	3.06	18.89	3.30
hin	6.67	1.58	6.44	1.55	9.56	2.40	6.67	1.47
hun	4.89	1.12	5.11	1.15	5.33	1.28	5.33	1.18
ice	9.56	2.11	9.78	2.14	10.22	2.21	10.00	2.36
kor	30.67	9.22	31.56	8.79	43.78	17.5	46.89	16.78
lit	18.67	3.53	20.00	3.93	20.67	3.65	19.11	3.55
ady	26.00	5.87	26.22	6.31	28.44	6.49	28.00	6.53
dut	16.00	2.92	15.78	2.86	15.78	2.89	16.44	2.94
jap	5.78	1.44	6.00	1.47	7.33	1.86	7.56	1.79
rum	10.44	2.35	10.89	2.41	12.00	2.62	10.67	2.53
vie	2.67	1.12	2.22	0.91	7.56	2.27	4.67	1.52
avg.	**14.52**	**3.24**	**14.96**	**3.31**	**17.51**	**4.30**	**16.84**	**3.99**

Table 4: Official Test results in WER and PER; CU-1=standard and multi-task transformer ensemble, CU-2=multi-task transformer ensemble, SIG-TB=SIGMORPHON transformer baseline, SIG-LSTM=SIGMORPHON LSTM baseline.

trained by us (an average of 5 models), while CU-1 and CU-2 are our submitted systems, which are described in Section 3.3. CU-1 performs best with an average performance of 14.43 WER and 3.33 PER, followed by CU-2 with 14.65 WER and 3.34 PER, respectively. Both CU-1 and CU-2 improve over the baseline for each of the 15 languages, with an average improvement of 1.82 WER and 1.6 WER, respectively. Both systems show an average improvement of 0.47 PER over the baseline, performing better on all languages, with the sole exception of Bulgarian, where the baseline slightly outperforms CU-2.

4.5 Official Shared Task Results

The results on the test set in Table 4 mirror our development set results. Our systems CU-1 and CU-2 are compared with the two best official baselines: a transformer (SIG-TB) and an LSTM sequence-to-sequence model (SIG-LSTM). CU-1 gives the best performance, with an average of 14.52 WER and 3.24 PER, followed by CU-2, with 14.96 WER and 3.31 PER. CU-1 shows an average improvement of 2.99 WER and 2.32 WER as well as 1.06 PER and 0.75 PER over SIG-TB and SIG-LSTM, respectively. CU-2 shows an average of 2.55 WER and 0.99 PER and, respectively, 1.88 WER and 0.68 PER improvement. Compared to all system submissions (Gorman et al.) CU-1 performs best on Lithuanian, with 18.67 WER and 3.53 PER. CU-2 performs best on Modern Greek, with 14.44 WER and 2.42 PER.

	T	T-E	MT	MT-E
arm	16.10	15.11	14.89	14.22
bul	32.06	28.22	31.82	30.22
fre	10.29	8.22	8.80	8.00
geo	26.03	25.78	25.29	24.67
gre	17.92	17.78	17.60	17.78
hin	8.78	6.67	7.20	6.89
hun	4.52	3.11	3.64	3.11
ice	12.20	10.00	10.53	9.11
kor	26.61	23.33	25.92	24.22
lit	22.88	21.56	23.02	22.44
ady	24.66	22.67	24.22	23.11
dut	16.80	15.33	15.11	14.44
jap	7.23	6.67	6.84	6.67
rum	13.04	12.89	12.31	12.00
vie	4.70	4.00	3.38	2.89
avg.	**16.25**	**14.76**	**15.37**	**14.65**

Table 5: Results of our ablation study in WER; T=standard transformer, T-E=standard transformer ensemble, MT=multi-task transformer, MT-E=multi-task transformer ensemble.

4.6 Ablation Study

We further perform an ablation study to explicitly investigate the impact of our two modifications – multi-task training and ensembling – with results shown in Table 5. T and MT are the standard and multi-task transformer, while T-E and MT-E are the ensembled versions of the same. The ensembles obtain better results: T-E shows an average improvement of 1.50 WER over T, and MT-E outperforms MT by 0.72 WER. Multi-task training also leads to performance gains, with MT improving over T by 0.88 WER and MT-E over T-E by 0.11 WER, showing that the effect of multi-task training is not as strong as that of ensembling. We conclude that both multi-task training and ensembling boost performance overall.

5 Conclusion

We described two CU Boulder submissions to SIGMORPHON 2020 Task 1. Our systems consisted of transformer models, some of which were trained in a multi-task fashion on G2P and P2G. We further created ensembles consisting of multiple individual models via majority voting.

Our internal experiments and the official results showed that these two straightforward extensions of the transformer model enabled our systems to improve over the official shared task baselines and a standard transformer model for G2P. Our final

models, CU-1 and CU-2, placed 6th and 8th out of 23 submissions, and obtained the best results of all systems for Lithuanian and Modern Greek, respectively.

Acknowledgments

We would like to thank all organizers of SIGMORPHON 2020 Task 1!

References

Alan W Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The Third ESCA Workshop in Speech Synthesis*. International Speech Communication Association.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

Noam Chomsky and Morris Halle. 1968. *The sound pattern of English*. Harper & Row New York.

Lucian Galescu and James F Allen. 2002. Pronunciation of proper names with a joint n-gram model for bi-directional grapheme-to-phoneme conversion. In *Seventh International Conference on Spoken Language Processing*.

Kyle Gorman, Lucas F.E. Ashby, Aaron Goyzueta, Arya D. McCarthy, Shijie Wu, and Daniel You. The sigmorphon 2020 shared task on multilingual grapheme-to-phoneme conversion.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.

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 Language Resources and Evaluation Conference*, pages 4216–4221, Marseille.

Benjamin Milde, Christoph Schmidt, and Joachim Köhler. 2017. Multitask sequence-to-sequence models for grapheme-to-phoneme conversion. In *INTERSPEECH*, pages 2536–2540.

Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using BERT. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5. IEEE.

Vincent Pagel, Kevin Lenzo, and Alan Black. 1998. Letter to sound rules for accented lexicon compression. *arXiv preprint cmp-lg/9808010*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

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.

Hao Sun, Xu Tan, Jun-Wei Gan, Hongzhi Liu, Sheng Zhao, Tao Qin, and Tie-Yan Liu. 2019. Token-level ensemble distillation for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1904.03446*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Robert Weide. 2005. The carnegie mellon pronouncing dictionary [cmudict. 0.6]. *Pittsburgh, PA: Carnegie Mellon University*.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2020. Applying the transformer to character-level transduction. *arXiv preprint arXiv:2005.10213*.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

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*.