Making a Point: Pointer-Generator Transformers for *Disjoint* Vocabularies

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

Abstract

Explicit mechanisms for copying have improved the performance of neural models for sequence-to-sequence tasks in the low-resource setting. However, they rely on an overlap between source and target vocabularies. Here, we propose a model that does not: a pointer-generator transformer for *disjoint* vocabularies. We apply our model to a low-resource version of the grapheme-to-phoneme conversion (G2P) task, and show that it outperforms a standard transformer by an average of 5.1 WER over 15 languages. While our model does not beat the the best performing baseline, we demonstrate that it provides complementary information to it: an oracle that combines the best outputs of the two models improves over the strongest baseline by 7.7 WER on average in the low-resource setting. In the high-resource setting, our model performs comparably to a standard transformer.

1 Introduction

Deep learning models define the state of the art on the majority of sequence-to-sequence tasks in natural language processing (NLP). Even when training data is limited, neural networks outperform many alternative approaches, e.g., on machine translation (Sennrich and Zhang, 2019) or on morphological generation tasks (Cotterell et al., 2018). For the second group, the use of mechanisms for copying has drastically improved performance when training sets are small (Cotterell et al., 2018).

Our work builds on the insight that the ability to copy elements from the input over to the output – as done by a pointer network (Vinyals et al., 2015) or a pointer-generator network (See et al., 2017) – can increase model performance on sequence-to-sequence tasks in the low-resource setting, as it simplifies the learning problem. However, existing neural models require that inputs and outputs consist of elements from overlapping sets. Here, we propose a **pointer-generator transformer model for *disjoint* source and target vocabularies**. Our model, shown in Figure 1, is a hybrid of an LSTM pointer-generator model (See et al., 2017) and a transformer model (Vaswani et al., 2017). Additionally, we integrate a mapping function, which defines a correspondence between elements in the source and target vocabularies.

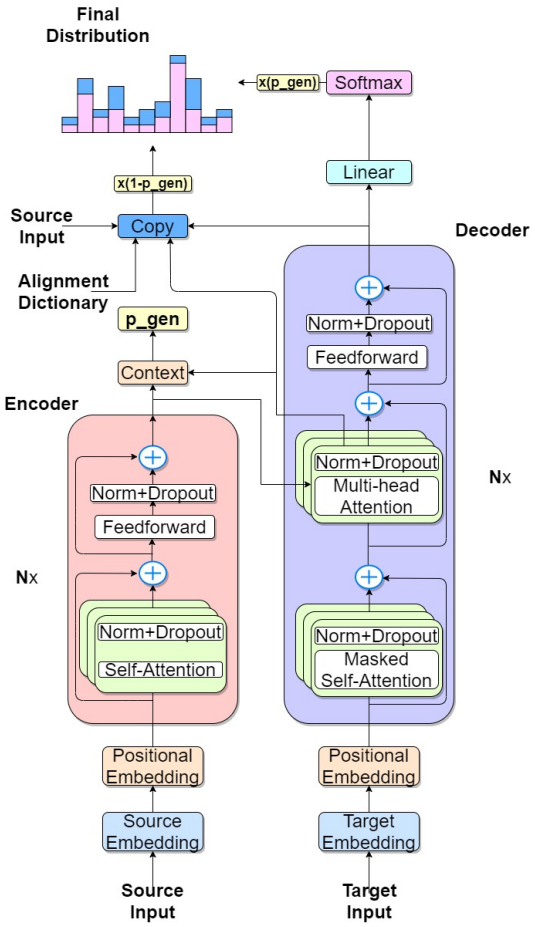

Figure 1: Architecture of our pointer-generator transformer for disjoint vocabularies.

We apply our model to the task of grapheme-to-phoneme conversion: mapping the spelling of

a word to a representation of its pronunciation (Bisani and Ney, 2008). This task is a great testing ground for our approach: input and output vocabularies are disjoint for many languages (cf. Table 1), and, as a character-level task with short sequences, it enables us to use small models, which allows for quick experimentation. G2P is also a task of high practical relevance: it is required for text-to-speech synthesis. Models that perform well in the low-resource setting will enable us to develop such language technologies for a wider set of languages.

We experiment with our model on varying training set sizes. In the low-resource setting, averaged over 15 languages, our architecture improves performance by 5.1 WER over a standard transformer. Further, it outperforms both the transformer and a copy baseline for up to 1000 training examples. While it underperforms our best performing baseline, we show that it provides valuable complementary information to it. In the high-resource setting, it performs comparably to a vanilla transformer.

2 Related Work

Sequence-to-sequence models. Popular neural architectures for sequence-to-sequence tasks include those based on LSTMs or GRUs in combination with attention (Bahdanau et al., 2015), transformer models, which use attention instead of recurrence (Vaswani et al., 2017), or pointer-generator models based on LSTMs (See et al., 2017). Sequence-to-sequence models have been applied to a large set of NLP tasks, including translation (Bahdanau et al., 2015; Vaswani et al., 2017), summarization (Raffel et al., 2019), morphological generation (Kann and Schütze, 2016), or historical text normalization (Flachs et al., 2019). To the best of our knowledge, pointer-generators have so far only been applied to tasks with overlappings source and target vocabularies (See et al., 2017; Sharma et al., 2018; Deaton et al., 2019). Here, we propose a pointer-generator transformer for tasks with *disjoint* vocabularies.

G2P. Early algorithms for G2P relied on handwritten parser-based rules in the format of Chomsky-Halle rewrite – or LTS – rules (Chomsky and Halle, 1968). Subsequently, other techniques have been developed, including rule-based systems (Black et al., 1998), maximum entropy models (Chen, 2003), LSTMs (Rao et al., 2015), or approaches based on semi-automatic alignment tables (Pagel et al., 1998). Our approach is similar to the idea of alignment tables, since we integrate a mapping function between vocabularies into a pointer-generator transformer. Today, neural sequence-to-sequence models are the standard approaches for G2P (Yao and Zweig, 2015; Sun et al., 2019; Gorman et al., 2020).

Makarov and Clematide (2020) proposed a model for G2P that, similar to our approach, makes use of explicit substitutions. Their BiLSTM-based neural transducer learns edit actions to transform an input sequence into a target sequence and is trained with imitation learning. Since this model is highly suitable for the low-resource setting, we compare our approach to it in our experiments.

	Low-Resource			High-Resource		
	Σ_{src}	Σ_{trg}	$\cap$	Σ_{src}	Σ_{trg}	$\cap$
arm	38	38	0	38	58	0
bul	29	40	0	30	67	0
fre	28	33	18	37	40	21
geo	31	32	0	33	35	0
gre	32	31	1	38	44	1
hin	49	48	0	60	88	0
hun	32	53	20	34	70	21
ice	31	53	19	36	77	23
kor	165	51	0	834	61	0
lit	31	64	18	32	110	20
ady	30	62	0	32	105	0
dut	27	40	18	35	50	21
jpn	65	47	0	78	79	0
rum	26	39	18	52	71	22
vie	75	47	15	93	49	18
avg.	**45.93**	**45.2**	**8.47**	**97.47**	**66.93**	**9.8**

Table 1: Number of tokens in the source and target vocabularies, and the number of shared tokens.

3 Pointer-Generator Transformers for Disjoint Vocabularies

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 2: The hyperparameters used in our experiments.

	Low-resource						High-resource					
	PG-T	T	LSTM	IM	Sub	O*	PG-T	T	LSTM	IM	Sub	O*
arm	62.2	70.5	72.2	**51.6**	57.3	38.2	15.6	**14.5**	14.7	14.9	57.3	10.7
bul	84.7	87.1	78.0	**73.6**	90.0	63.6	33.1	31.9	31.1	**29.8**	86.9	18.7
fre	87.1	88.7	93.1	**68.9**	95.3	61.3	7.3	8.0	**6.2**	7.6	95.3	4.7
geo	65.6	77.8	60.2	62.0	**50.7**	45.1	**25.6**	27.8	26.4	26.9	44.7	19.3
gre	77.6	81.1	84.2	**44.0**	74.2	39.6	**17.1**	18.0	18.9	18.2	71.1	11.1
hin	**70.7**	82.6	80.4	78.0	72.7	60.2	8.0	8.2	**6.7**	6.9	69.8	3.8
hun	74.0	83.2	83.1	**41.1**	56.2	36.9	6.7	5.7	5.3	**4.4**	54.9	3.1
ice	87.1	90.9	94.2	**60.0**	85.3	56.0	11.1	10.6	**10.0**	11.6	85.3	8.2
kor	99.6	98.9	100.0	**75.8**	100.0	75.6	35.1	33.5	46.9	**28.7**	100.0	20.2
lit	87.1	89.6	97.1	**58.7**	97.6	52.9	22.2	21.0	19.1	**18.2**	96.9	16.2
ady	87.1	90.8	89.3	**64.0**	90.9	58.9	**27.1**	27.6	28.0	30.4	91.3	22.5
dut	90.9	93.0	93.3	**61.1**	96.2	58.5	18.7	18.2	**16.4**	19.8	97.1	12.5
jap	77.8	85.7	97.3	**76.2**	99.8	64.0	7.3	7.4	7.6	**7.1**	99.6	5.3
rum	64.0	73.5	59.3	**33.1**	48.9	29.1	13.6	12.1	**10.7**	13.8	51.3	10.9
vie	90.0	88.3	99.6	**82.7**	100.0	75.8	4.4	3.6	4.7	**1.1**	100.0	1.1
Avg.	80.4	85.5	85.4	**62.1**	81.0	54.4	16.9	16.5	16.9	**16.0**	80.1	11.2

Table 3: Test set results for WER; all models are described in the text. The best performance (excluding the oracle) is shown in bold.

3.1 Model Architecture

Our model, cf. Figure 1, is a hybrid of a transformer (Vaswani et al., 2017) and a pointer-generator network (See et al., 2017) with a separate mapping function between vocabularies. Like the transformer, it is an auto-regressive encoder-decoder architecture with stacked self-attention and fully connected encoder and decoder layers. The decoder, in addition, employs multi-head attention over the encoder outputs. We further add a component which outputs the probability of generating a token, as opposed to copying an element from the source. Following See et al. (2017), the generation probability $p_{gen} \in [0, 1]$ at time step t is computed as:

$$p_{gen} = \sigma(w_c^T c_t + w_s^T s_t + w_x^T x_t + b_{ptr}) \quad (1)$$

c_t represents the context vector at step t, which is the sum of the embedded encoder hidden states h, weighted by the multi-head attention weights a: $c_t = \sum_i a_i^t h_i$. s_t and x_t are the hidden respresentation in the last decoder layer and, respectively, the decoder input. $w_{h^*}^T$, w_s^T, w_x^T are learned weights and b_{ptr} is a learned bias vector.

We also account for the fact that our architecture should handle *disjoint* source and target vocabularies, denoted as as Σ_{src} and Σ_{trg}. To enable the use of a pointer mechanism across vocabularies, we define a mapping function m from the source vocabulary to the target vocabulary:

$$m : \Sigma_{src} \rightarrow \Sigma_{trg} \quad (2)$$

Our model then computes the probability of an output character $k \in \Sigma_{trg}$ for an input sequence $g \in \Sigma_{src}^*$ at each time step t as:

$$P(k) = p_{gen} P_{\Sigma_{trg}}(k) + (1 - p_{gen}) \sum_{i:k=m(g_i)} a_i^t \quad (3)$$

$P_{\Sigma_{trg}}(k)$ represents the probability of k to be generated by the decoder in the standard way, given the input sequence and previously generated tokens. It is weighted by the generation probability p_{gen}. The second term represents the attention over the encoder outputs, which is multiplied by $1 - p_{gen}$, the copy probability. The target indices receiving probability are found with the mapping m.

3.2 Source-to-Target Vocabulary Mapping

We obtain the mapping function m as follows.

Alignment. We first compute an alignment between source and target tokens in the training files for all languages and settings with the GIZA++ aligner (Och and Ney, 2003), employing the default parameters. The alignment is computed by a hidden Markov model.

Mapping. We then construct a mapping between characters in the source and target vocabularies by assigning the target token with the highest type-level alignment probability to each source token. The mapping function m can map multiple source tokens to the same target token. We create separate mapping functions for all languages and settings in our experiments as they change for different training sets.

	Low-resource					High-resource				
	PG-T	T	LSTM	IM	Sub	PG-T	T	LSTM	IM	Sub
arm	19.0	23.6	21.3	14.4	**12.3**	3.9	3.5	3.5	**3.3**	12.1
bul	30.9	33.6	23.1	**19.2**	25.6	7.1	6.5	5.9	**5.7**	25.7
fre	40.2	44.1	41.9	**22.0**	53.7	1.9	2.0	**1.3**	1.8	52.8
geo	18.9	24.0	13.3	11.8	**8.7**	4.8	5.2	5.1	**4.6**	8.2
gre	26.7	29.2	24.0	**10.3**	18.5	**2.8**	2.9	3.3	3.2	18.0
hin	26.9	35.7	31.6	29.2	**23.4**	2.0	2.0	**1.5**	1.4	22.1
hun	24.6	33.1	27.4	**10.0**	17.8	1.4	1.3	**1.2**	**1.2**	17.5
ice	43.0	44.7	42.7	**17.9**	30.4	2.8	**2.4**	**2.4**	2.5	29.6
kor	65.1	69.0	78.3	**23.8**	52.7	10.4	9.9	16.8	**4.7**	50.4
lit	39.5	43.2	53.1	**8.8**	37.4	4.2	3.9	3.6	**2.9**	36.9
ady	37.3	42.1	33.1	**18.2**	46.9	6.8	6.6	**6.5**	7.2	44.8
dut	42.6	44.5	40.8	**15.1**	37.0	3.6	3.6	**2.9**	3.7	37.0
jap	34.1	41.4	54.6	**26.1**	48.4	1.9	2.2	1.8	**1.7**	50.6
rum	21.4	26.0	14.1	**8.5**	11.4	3.1	2.7	**2.5**	3.7	10.7
vie	44.5	45.1	65.1	**27.2**	65.4	1.5	1.6	1.5	**0.3**	58.8
Avg.	34.3	38.6	37.6	**17.5**	32.6	3.9	3.8	4.0	**3.2**	31.7

Table 4: Test set results for PER; all models are described in the text. The best performance is shown in bold.

Our system is built on the transformer implementation by Wu et al. (2020), and our final code is available on github.[1]

4 Experiments

4.1 Data, Metrics, and Baselines

Data. We use the G2P data from Gorman et al. (2020), which covers a set of 15 typologically diverse languages. We construct our low-resource experiments by taking the first 100 instances from each training set. For the high-resource experiments, we leverage all available data. Development and test sets are the same in both settings. Table 1 shows the number of characters in Σ_{src} and Σ_{trg} according to the training set, as well as the number of characters shared between both. We can see that, for many languages, the overlap is 0, i.e., the vocabularies are disjoint sets.

Metrics. We use word error rate (WER), i.e., the percentage of words that are correct, as our main metric. We also measure the phoneme error rate (PER), i.e., the percentage of correctly generated phonemes. We use the SIGMORPHON 2020 Task 1 (Gorman et al., 2020) evaluation script[2] to calculate these.

Baselines. We compare our model (**PG-T**) to four baselines: **T** is a vanilla transformer with hyperparameters identical to those in PG-T for the high-resource setting (following Gorman et al. (2020); listed in Table 2). For the low-resource setting, all hyperparameters remain the same, except the batch size which is set to 32. Thus, T corresponds to our model, but without the pointing mechanism. **IM** is the neural transducer trained with imitation learning detailed in Section 2 (Makarov and Clematide, 2020). **LSTM** is the LSTM encoder-decoder baseline of SIGMORPHON 2020 Task 1 (Gorman et al., 2020). The output of **Sub** is the sequence of target tokens corresponding to each source token, according to m. Finally, **O*** is not a baseline but an oracle model that combines the best outputs of **PG-T** and **IM**.

4.2 Results

Low-resource G2P. Table 3 shows the WER of our model and all baselines on the test set. In the low-resource setting, PG-T outperforms all baselines except for IM, with an average increase of 0.6 over Sub, the second best baseline after IM. It further improves over T by 5.1 WER, which shows the effectiveness of our pointer extension. While the average improvement over Sub is modest, PG-T shows large performance gains over Sub for individual languages such as Japanese. Similarly, for Hindi, our model outperforms IM by 7.3 WER.

High-resource G2P. In the high-resource setting, PG-T obtains an average performance of 16.9 WER. T and IM slightly outperform PG-T: T by 0.4 WER and IM by 0.9 WER. PG-T and LSTM reach the same average WER. Sub improves minimally as compared to the low-resource setting and is vastly outperformed by all other approaches. We show PER results for the low-resource and the high-

[1] https://github.com/nala-cub/g2p-PG-T

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

resource setting in Table 4. All development set results for both WER and PER can be found in the appendix.

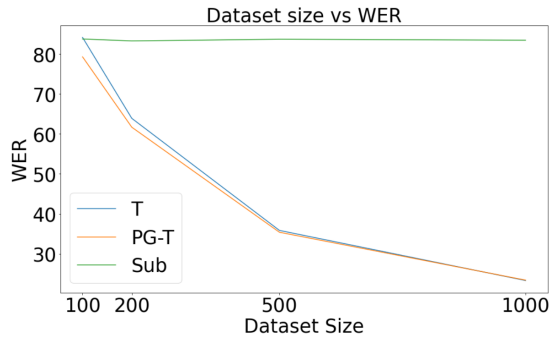

Figure 2: Performance for increasing amounts of training data on the development set.

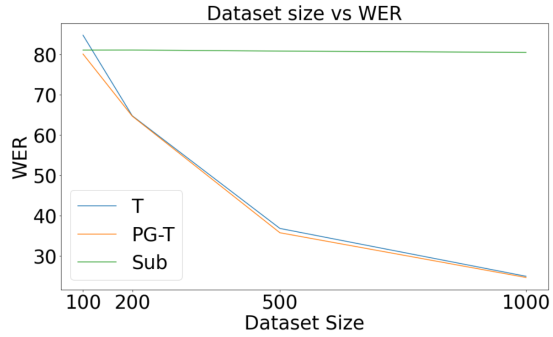

Figure 3: Performance for increasing amounts of training data on the test set.

Learning curve. We also look at the learning curves for increasing dataset sizes (Figures 2, 3) and compare our model to the two baselines T and Sub, whose main ideas PG-T combines. Compared to T, PG-T shows the biggest improvements for 100 training examples. The performance difference between the two gets smaller as the size of the dataset increases. However, for 1000 training examples, PG-T still performs slightly better than T. With regards to Sub, our model outperforms it slightly for 100 examples, but this gap widens very quickly, due to Sub's inability to learn much from the training data. Overall, we show that, over a varied amount of data in the low-resource setting, our model outperforms both baselines and, thus, that our combination of the two is effective.

While our model does not perform as well as IM, our strongest baseline, we demonstrate that it provides valuable complementary information. In both the low-resource and the high-resource setting and across all languages, an oracle combination of PG-T with IM yields better results than either model on its own. We will discuss this in the next section.

	CCO	UCO-PG-T	UCO-IM	CMO
arm	115	69	78	127
bul	40	23	146	52
fre	36	42	122	43
geo	91	59	124	118
gre	64	28	158	77
hin	48	79	54	53
hun	122	22	175	131
ice	47	17	132	56
kor	2	2	104	3
lit	33	15	168	39
ady	37	18	128	45
dut	26	10	158	30
jpn	40	51	61	43
rum	150	16	144	165
vie	21	34	77	22
Avg.	**58.13**	**32.33**	**121.93**	**66.9**

Table 5: Output comparison of PG-T and IM for the development set in the low-resource setting.

	CCO	UCO-PG-T	UCO-IM	CMO
arm	354	21	19	397
bul	257	43	34	294
fre	385	16	19	401
geo	299	38	35	365
gre	349	19	34	385
hin	403	14	15	418
hun	425	5	11	431
ice	384	13	20	403
kor	288	28	69	339
lit	326	14	38	364
ady	313	28	26	376
dut	346	28	27	378
jpn	410	7	8	430
rum	380	11	14	419
vie	422	5	19	425
Avg.	**356.1**	**19.3**	**25.87**	**388.33**

Table 6: Output comparison of PG-T and IM for the development set in the high-resource setting.

5 Analysis

5.1 Oracle

O* in Table 3 is a hypothetical oracle that would be the result of combining the best outputs of PG-T and IM. The results in Table 3 show that the oracle performs far better than IM, by 7.7 WER and 4.8 WER in the low-resource and the high-resource setting, respectively. This indicates that there are differences in the behaviour of the two models, and that they provide complementary information. We will discuss the differences in the models' predictions next.

5.2 Output Comparisons

In this subsection, we go over the output comparisons in Tables 5 and 6. In those tables, we compare the phoneme outputs generated by PG-T and IM. In particular, we look at the number of (1) correct outputs both models have in common (CCO), (2) correct outputs that are unique to each model (UCO-PG-T and UCO-IM), and (3) outputs both models have in common (CMO), independent of if they are correct or incorrect.

Tables 5 and 6 compare the outputs of PG-T and IM in the low-resource and the high-resource setting, respectively. The average CCO count is 58.13 in the low-resource setting and 356.1 in the high-resource setting (out of a total of 450 data points), showing that part of what is learned is shared between both models. However, in the high-resource setting, the average UCO counts are 19.3 for UCO-PG-T and 25.87 for UCO-IM, respectively. This indicates that, while their performances are similar, the models are learning some complementary information. This is reflected in the low-resource setting as well: even though IM performs better on average, PG-T still learns relevant complementary information, with an average of 32.33 UCO-PG-T. We find the same for the individual languages in our experiments.

5.3 Error Analysis

As a case study, we further perform an analysis of the errors on the Hindi development set in the high-resource setting.[3] For this particular combination of language and dataset size, LSTM performs best with 4.7 WER, followed by IM with 7.1 WER, PG-T with 7.3 WER, and T with 8.6 WER. Sub performs badly with 68.2 WER. Almost all errors generated by the high-performing models (IM, LSTM, PG-T and T) center around the halant[4] morpheme in the Hindi script, which indicates the lack of an inherent vowel following a consonant. Its phoneme complement is the schwa[5], which indicates an unstressed vowel. LSTM and T behave identically on most examples, while PG-T often behaves in the opposite way, i.e., in cases where LSTM generates a schwa, PG-T does not, and vice versa. PG-T and IM, however, have an even split among examples where either model makes a schwa-based error.

[3] WER and PER of all models on the development sets for all languages are shown in Table 7 in the appendix.

[4] https://en.wiktionary.org/wiki/halant

[5] https://en.wiktionary.org/wiki/schwa

6 Conclusion

We introduced a pointer-generator transformer for sequence-to-sequence tasks with disjoint vocabularies, and evaluated it on G2P in 15 different languages. While our model did not perform as well as our strongest baseline, we showed that our model learns complementary information to the latter, thus succeeding on examples the baseline fails on. We further demonstrated that combining the best outputs of our pointer-generator transformer and the baseline results in a lower WER on low-resource G2P than each individual model. We leave the development of a system that can combine the two G2P systems for future work.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

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 Workshop in Speech Synthesis*. International Speech Communication Association.

Stanley F Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Eighth European Conference on Speech Communication and Technology*.

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

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels. Association for Computational Linguistics.

Jon Deaton, Austin Jacobs, Kathleen Kenealy, and Abigail See. 2019. Transformers and pointer-generator networks for abstractive summarization.

Simon Flachs, Marcel Bollmann, and Anders Søgaard. 2019. Historical text normalization with delayed rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1614–1619, Florence, Italy. Association for Computational Linguistics.

Kyle Gorman, Lucas F.E. 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*. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany. Association for Computational Linguistics.

Peter Makarov and Simon Clematide. 2020. Cluzh at 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 171–176.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. *arXiv preprint arXiv:1905.11901*.

Abhishek Sharma, Ganesh Katrapati, and Dipti Misra Sharma. 2018. IIT(BHU)–IIITH at CoNLL–SIGMORPHON 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 105–111, Brussels. Association for Computational Linguistics.

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.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

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*.

Appendix B: Development Set Results

	Low-resource						High-resource					
	PG-T	T	LSTM	IM	Sub	O*	PG-T	T	LSTM	IM	Sub	O*
arm	59.1	70.7	74.0	57.1	**56.2**	41.8	16.5	16.0	14.9	17.1	**56.7**	3.8
bul	86.0	88.4	84.7	**58.7**	92.7	53.6	33.3	32.3	**28.4**	35.3	90.4	25.8
fre	82.7	86.6	92.7	**64.9**	97.6	55.6	10.9	10.2	**7.1**	10.2	97.3	6.7
geo	66.7	76.4	63.6	**52.2**	56.4	39.1	25.1	26.1	**21.1**	25.8	51.1	17.3
gre	79.6	82.4	82.7	**50.7**	75.3	44.5	18.2	18.1	**13.6**	14.9	74.7	10.7
hin	**71.8**	78.7	79.8	77.3	72.2	59.8	7.3	8.6	**4.7**	7.1	68.2	4.0
hun	68.0	81.2	80.9	**34.0**	75.1	29.1	4.4	4.4	3.3	**3.1**	74.7	2.0
ice	85.8	89.1	93.3	**60.2**	93.1	56.5	11.8	11.8	**9.3**	10.2	93.1	7.3
kor	99.1	99.4	100.0	**76.4**	100.0	76.0	29.8	26.5	41.1	**20.7**	100.0	14.5
lit	89.3	90.2	97.8	**55.3**	98.0	52.0	24.4	22.7	**16.9**	19.1	98.0	16.0
ady	87.8	88.0	88.7	**63.3**	91.3	59.3	24.2	24.6	**22.7**	24.7	90.4	18.5
dut	92.0	92.4	94.4	**59.1**	96.4	56.9	16.9	16.6	**12.2**	17.1	99.1	10.9
jap	79.8	85.0	94.7	**77.6**	99.3	66.2	7.3	7.0	**6.7**	7.1	100.0	5.6
rum	63.1	70.1	61.3	**34.7**	52.9	31.1	13.1	12.6	**9.3**	12.4	54.9	10.0
vie	87.8	88.1	98.4	**78.2**	100.0	70.7	5.1	4.5	4.2	**2.0**	100.0	0.9
Avg.	79.9	84.5	85.8	**60.0**	83.8	52.8	16.6	16.1	**14.4**	15.1	83.2	10.3

Table 7: Development set results for WER; all models are described in the text. The best performance (excluding the oracle) is shown in bold.

	Low-resource					High-resource				
	PG-T	T	LSTM	IM	Sub	PG-T	T	LSTM	IM	Sub
arm	18.4	23.5	22.5	14.2	**12.6**	3.4	3.4	**2.9**	3.4	12.6
bul	31.4	35.4	26.4	**15.9**	25.4	7.6	7.2	**6.2**	7.3	25.1
fre	40.9	44.8	44.6	**21.6**	58.2	2.8	2.6	**1.8**	2.9	57.5
geo	19.8	23.8	14.7	11.6	**10.1**	4.9	5.2	**4.5**	4.6	9.8
gre	27.5	29.3	24.6	**11.1**	20.4	3.6	3.4	**2.7**	**2.7**	20.6
hin	28.6	35.6	32.8	29.0	**23.3**	2.1	2.3	**1.4**	1.6	22.0
hun	23.3	32.5	26.8	**8.4**	22.7	0.9	1.0	**0.6**	0.7	22.4
ice	44.1	44.8	42.6	**17.6**	33.5	3.1	2.8	**2.0**	2.4	31.1
kor	68.0	72.2	78.5	**24.3**	53.1	7.6	7.6	16.6	**3.7**	51.2
lit	40.7	42.4	56.0	**8.7**	46.6	4.6	4.4	3.4	**3.1**	45.7
ady	37.9	41.9	32.6	**19.9**	48.0	6.0	6.4	**5.7**	5.9	45.0
dut	42.3	45.4	41.7	**14.7**	39.7	3.4	3.5	**2.1**	3.4	43.0
jap	34.0	40.0	55.0	**28.8**	48.4	2.5	2.4	2.1	**1.9**	49.8
rum	21.1	25.4	15.0	**9.2**	12.4	3.3	3.3	**2.3**	3.1	11.9
vie	46.5	44.4	63.8	**26.7**	66.9	1.5	1.6	1.3	**0.4**	63.7
Avg.	35.0	38.8	38.5	**17.5**	34.8	3.8	3.8	3.7	**3.1**	34.1

Table 8: Development set results for PER; all models are described in the text. The best performance is shown in bold.