IIT(BHU)–IIITH at CoNLL–SIGMORPHON 2018 Shared Task on Universal Morphological Reinflection

Abhishek Sharma [*]
IIT (BHU), Varanasi

Ganesh Katrapati
Language Technologies Research Center
IIIT Hyderabad

Dipti Misra Sharma
Language Technologies Research Center
IIIT Hyderabad

Abstract

This paper describes the systems submitted by IIT (BHU), Varanasi/IIIT Hyderabad (IITBHU–IIITH) for Task 1 of CoNLL–SIGMORPHON 2018 Shared Task on Universal Morphological Reinflection (Cotterell et al., 2018). The task is to generate the inflected form given a lemma and set of morphological features. The systems are evaluated on over 100 distinct languages and three different resource settings (low, medium and high). We formulate the task as a sequence to sequence learning problem. As most of the characters in inflected form are copied from the lemma, we use Pointer-Generator Network (See et al., 2017) which makes it easier for the system to copy characters from the lemma. Pointer-Generator Network also helps in dealing with out-of-vocabulary characters during inference. Our best performing system stood 4th among 28 systems, 3rd among 23 systems and 4th among 23 systems for the low, medium and high resource setting respectively.

1 Introduction

Morphological Inflection is the process of inflecting a lemma according to a set of morphological features so that the lemma becomes in accordance with other words in the sentence. It is useful for alleviating data sparsity, especially in morphologically rich languages during Natural Language Generation. For example, Minkov et al. (2007) translate words from the source language to lemmas in the target language and then use Morphological Inflection as a post-processing step to make the words of the output sentence in agreement with each other. Not only their approach reduces the data sparsity by decreasing the number of candidate words while translating, it also gives better results.

[*]This research was conducted during the authors internship at IIIT Hyderabad.

CoNLL–SIGMORPHON 2018 Shared Task on Universal Morphological Reinflection consisted of two tasks. Participants could compete in either or both of the tasks. We participated in Task 1 only. The task was to build a system which could inflect a lemma given a set of morphological tags. The systems were evaluated on over 100 distinct languages, out of which 10 were surprise languages. An example showing input and the expected output of the system is given below.

$$\text{(touch, V;V.PTCP;PRS)} \rightarrow \text{touching}$$

To assess the system's ability to generalize in different resource settings, three varying amounts of labeled training data (low, medium, high) were given. The systems were evaluated separately for each language and the three data quantity conditions. Accuracy (the fraction of correctly predicted forms) and the average Levenshtein distance between the prediction and the truth across all predictions were used as metrics. An aggregated performance measure separate for each of the resource setting was obtained by averaging the results for individual languages.

Morphological Inflection is accomplished by different morphological processes such as prefixation, infixation, suffixation (attaching bound morpheme in front, within and at the end of stem respectively) and ablaut depending on the language. As the systems were evaluated on over 100 distinct languages, we were motivated to use neural network based approaches because they do not require any manual feature engineering. But neural networks require a lot of training data to work. We try to address this challenge by designing neural network architectures which work well even on the low resource setting of the task.

Our system is based on attention based encoder-decoder models (Bahdanau et al., 2014). The

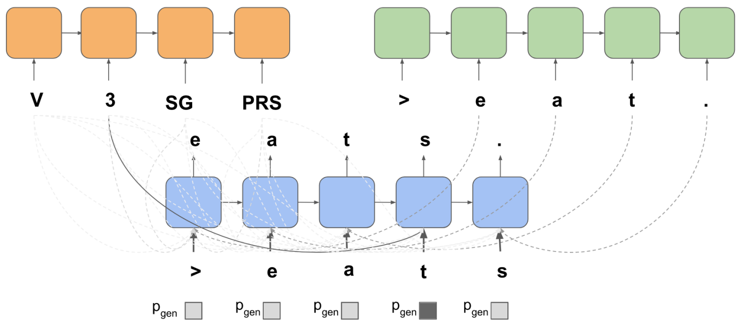

Figure 1: Neural network architecture for our system. The two encoders are shown at the top, while the decoder is shown at the bottom. At each time step, the decoder computes attention distribution over both the lemma and the tags separately. Attention mechanism is shown by the dotted lines (darker colour corresponds to more weight). A scalar - generation probability $p_{gen} \in [0, 1]$ (shown as the square, the lighter the colour the less the value) is also calculated at each time step, which corresponds to how likely a character will be generated from the vocabulary instead of a character being copied from the lemma.

lemma and the tags are encoded using two separate encoders. While decoding, the decoder reads relevant parts of the lemma and the tags using attention mechanism. As most of the characters in the inflected form are copied from the lemma, it is necessary to design a system with strong tendency to copy. We use Pointer-Generator Network (See et al., 2017) which facilitates copying of characters of lemma and tackles the problem of out-of-vocabulary tokens during prediction. Compared to other similar performing systems, our system is trained end-to-end, doesn't require data augmentation techniques and uses soft attention over hard monotonic attention which makes it more flexible.

Our best performing system outperforms the baseline by 14.21%, 22.41% and 19.13% for the low, medium and high resource settings respectively. It stood 4th among 28 systems, 3rd among 23 systems and 4th among 23 systems for the low, medium and high data conditions respectively.

The remainder of this paper is organized as follows. We present prior work on Morphological Inflection in Section 2. We describe our system in Section 3. The results of the shared task are presented in Section 4. In Section 5, we present ablation studies and discuss the contribution of the specific design decisions we made to the performance of our systems. We conclude the paper with Section 6.

2 Background

Traditional approaches for morphological inflection involve crafting hand-engineered rules. Although these rules offer high accuracy, they are very expensive to create.

Machine learning based approaches treat morphological inflection as a string transduction task (Durrett and DeNero, 2013; Hulden et al., 2014; Ahlberg et al., 2015; Nicolai et al., 2015). These approaches extract rules automatically from the data, but they still require language specific feature engineering.

Neural network based approaches successfully solve this problem. These approaches require no feature engineering and the same architecture works for different languages. Faruqui et al. (2016) were the first to formulate morphological inflection as neural sequence to sequence learning problem (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014). Kann and Schütze (2016) improved on their approach by using a single model instead of separate models for each morphological feature. They fed morphological tags into the encoder along with the sequence of characters of lemma. They also used attention mechanism (Bahdanau et al., 2014). Aharoni and Goldberg (2017) present an alternative to the soft attention in form of hard monotonic attention which models the almost monotonic alignment between characters in lemma and the inflected form.

The best performing system (Makarov et al., 2017) of the previous edition of this shared task extended the hard monotonic attention model of Aharoni and Goldberg (2017) with a copy mechanism (HACM model). To deal with low training data especially in the low and medium resource settings, some teams used data augmentation techniques (Kann and Schütze, 2017; Bergmanis et al., 2017; Silfverberg et al., 2017; Zhou and Neubig, 2017; Nicolai et al., 2017).

3 System Description

In this section, we describe our system in detail. We report the neural network architecture, the training process, the hyperparameters and our submissions.

3.1 Neural network architecture

Our neural network architecture is based on Pointer-Generator Network (See et al., 2017) with some subtle differences.

Characters of the lemma c_i along with the additional start and stop characters are fed one by one into a bidirectional LSTM encoder producing a sequence of hidden states h_{l_i}. Similarly, using a separate bidirectional LSTM encoder, the tags tg_i are encoded and another sequence of hidden states h_{tg_i} is obtained.

We use a unidirectional LSTM as the decoder. The decoder's hidden state s_i is initialised by applying an affine transformation on the concatenation of the last hidden states of the lemma and the tag encoders. As the input and output sequences have different semantics, this affine transformation gives the model the ability to learn transformation of semantics from input to output (Faruqui et al., 2016).

$$s_0 = W_{initial}[h_{l_N}; h_{tg_N}] + b \tag{1}$$

While decoding, at each time step t, the decoder computes an attention distribution over the lemma and the tag separately denoted as a_l^t and a_{tg}^t (Bahdanau et al., 2014).

$$e_{l_i}^t = v^T \tanh(W_{h_l} h_{l_i} + W_{s_l} s_{t-1} + b_l) \tag{2}$$

$$e_{tg_i}^t = v^T \tanh(W_{h_{tg}} h_{tg_i} + W_{s_{tg}} s_{t-1} + b_{tg}) \tag{3}$$

$$a_l^t = \text{softmax}(e_l^t) \tag{4}$$

$$a_{tg}^t = \text{softmax}(e_{tg}^t) \tag{5}$$

The context vectors h_l^* and h_{tg}^* are computed as the weighted sum over the encoder hidden states h_{l_i} and h_{tg_i} with the attention distribution mass a_l^t and a_{tg}^t as weights.

$$h_{l_t}^* = \sum_i a_{l_i}^t h_{l_i} \tag{6}$$

$$h_{tg_t}^* = \sum_i a_{tg_i}^t h_{tg_i} \tag{7}$$

The combined context vector is obtained by simply concatenating the lemma and the tag context vector.

$$h_t^* = [h_{l_t}^*; h_{tg_t}^*] \tag{8}$$

The combined context vector h_t^* and the embedding of character predicted at the previous time step, y_{t-1} (while training to speed up convergence we use the ground truth label y_{t-1}^* instead) is given as input to the decoder. At the first time step, start character is given as input in place of y_{t-1}.

$$s_t = f(s_{t-1}, h_t^*, y_{t-1}) \tag{9}$$

where f is a nonlinear function.

A probability distribution over the characters in the vocabulary is calculated which corresponds to how likely will a particular character be generated (if a character is generated at all).

$$P_{vocab} = \text{softmax}(V[s_t; h^*] + b) \tag{10}$$

At each time step, a generation probability $p_{gen} \in [0, 1]$ is calculated. The generation probability determines if the decoder will generate a character from the vocabulary or copy a character from the lemma.

$$p_{gen} = \sigma(w_h h_t^* + w_s s_t + w_y y_{t-1} + b) \tag{11}$$

Note that here p_{gen} is calculated using y_{t-1} (the embedding of output produced at the previous time step) instead of the decoder input x_t as in See et al. (2017).

The probability of predicting a character c is computed as the sum of probability of generating c weighted by the generation probability p_{gen} and the total attention distribution over c weighted by the probability of copying it $(1 - p_{gen})$.

$$P(c) = p_{gen} P_{vocab}(c) + (1 - p_{gen}) \sum_{i:c_i=c} a_{l_i}^t \tag{12}$$

The decoder keeps predicting characters until the stop character is predicted or a fixed number of time steps are reached.

3.2 Training

We use negative log likelihood to compute the loss. The loss for time step t, where c_t^* is the target character is given by,

$$loss_t = -\log P(c_t^*) \tag{13}$$

	low	medium	high
embedding size	100	100	300
hidden units	100	100	100
dropout probability (p)	0.5	0.5	0.3
initial epochs (e_1)	300	80	60
extended epochs (e_2)	100	20	10

Table 1: Hyperparameters for low, medium and high resource settings.

The loss for the overall sequence is,

$$loss = \frac{1}{T}\sum_{t=0}^{T} loss_t \tag{14}$$

We use Adam Optimiser (Kingma and Ba, 2014) with initial learning rate 0.001 and batch size 32 to train the neural network. To deal with exploding gradient problem, we clip the norm computed over all the gradients together to 3. We apply dropout (Srivastava et al., 2014) with probability p over embeddings and the encoder hidden states.

We use early stopping to prevent overfitting. A portion of the development set is used as the validation set. After each epoch, performance on validation set is calculated. Initially the model is trained on e_1 epochs. If the highest performance on validation set is obtained within e_2 recent epochs, the model is further trained for e_2 epochs. This goes on until performance on validation set stops improving.

Single layer LSTMs were used as encoders and decoders to reduce number of parameters. Optimal size of embeddings and the number of hidden units in LSTMs were determined based on the performance of the model on a subset of languages in development set.

The values for hyperparameters p, e_1, e_2, embedding size and hidden units of LSTM are given in Table 1.

We used PyTorch for implementing the network. The code for the system is available at `https://github.com/abhishek0318/conll-sigmorphon-2018`.

3.3 Submissions

We made a total of two submissions. For the first submission, we trained only one system for each language and data resource setting pair. We used ensembling technique for the second submission. We trained 5, 3 and 1 system(s) for each language in low, medium and high data resource settings respectively. Their predictions were combined using hard voting.

	System 1	System 2	Baseline
low	49.79%	**52.60%**	38.20%
medium	82.90%	**84.19%**	61.78%
high	94.43%	**94.43%**	75.30%

Table 2: Average accuracy of the system on the test set over all the languages for low, medium and high resource settings respectively.

4 Results

Average accuracy of the system over all the languages in a data resource setting is presented in Table 2.

Our best performing system outperforms the baseline by very large margins - 14.21%, 22.41% and 19.13% for the low, medium and high resource settings respectively.

We observe that using ensembling technique (in the second submission) gives a boost of few percentage points in the accuracy over the first submission, where ensembling is not used.

5 Ablation Studies

In this section, we investigate how difference system design choices influenced the performance of the system. As reasonable performances were obtained for medium and high resource settings in previous editions of the shared task, we focus our attention to the low resource setting and compare models on this setting.

5.1 Pointer Generator Network

We examine the performance gain obtained by using Pointer-Generator Network, the essence of our system. We compare the performance of a simple attention based neural encoder-decoder model with and without using ideas from Pointer-Generator Network.

Consider the architecture proposed by Kann and Schütze (2016) for the task of morphological inflection. The architecture is based on simple attention based encoder-decoder model. The source sequence s_i consists of the characters of the lemma followed by the tags.

We include ideas from Pointer-Generator Network into this model. At each time step, the decoder calculates generation probability p_{gen} (See et al., 2017). The network uses the computed attention distribution to determine which character from the lemma it should copy. Because there is only a single encoder, the attention distribution is over both the lemma and the tags. The tags therefore have some attention over them. To use Equation 12, we must normalise the attention weights of the characters, so that we have a new attention distribution over the set of characters.

$$P(c) = p_{gen}P_{vocab}(c) + (1 - p_{gen})\frac{\sum_{i:s_i=c} a_i^t}{\sum_{i:s_i \in C} a_i^t} \tag{15}$$

We use modified form of Equation 12 as shown above to calculate $P(c)$. Here C is the set of characters.

For the same hyperparameters, the architecture used in Kann and Schütze (2016) gives 21.99% average accuracy as compared to the architecture including ideas from Pointer-Generator Network, which gives 44.02% average accuracy tested on development set over all the languages for low resource setting. Thus using Pointer Generator Network increases the performance of the system tremendously for low resource setting.

5.2 Separate Encoder for Tags

We investigate the benefit of using a separate encoder for the tags, instead of encoding them using a same encoder as in Kann and Schütze (2016).

Consider the neural network architecture with two separate encoders for the lemma and the tags. At each timestep while decoding, attention distribution is computed over the lemma. The last hidden state of the tag encoder is used as the representation of the set of tags. It along with the context vector of the lemma is fed to the decoder at each time step. We compare the performance of this architecture, to the architecture described in Section 5.1 (which uses single encoder for the lemma and tags and Pointer-Generator Network). The architecture with a single encoder obtains 44.02% average accuracy, while the one with two separate encoders achieves 48.18% average accuracy tested on the development set for low resource setting.

A possible explanation for the difference in the performance is that the lemma and the tags are completely separate entities and a single encoder can't encode them correctly. We were motivated to represent the tags using embeddings as embeddings have more representational power compared to zeros and ones in case of one hot encoding. As the number of tags vary for each example, using LSTM to encode them seemed apt. Note that the representation obtained using this approach is not order invariant. Using order invariant representations (Vinyals et al., 2016; Zaheer et al., 2017) is left as future work.

5.3 Attention over Tags

We inspect whether using attention over the sequence of tags as compared to using a fixed vector representation gives better results. We consider the architecture introduced in Section 5.2. Instead of using last hidden state of the encoder to represent the tags, we use attention over tags too and compare the performance. Note this is same architecture we described in 3.1. Using attention over tags leads to average accuracy of 49.08% as compared to 48.18% on the development set for low resource setting.

This can explained as by using attention mechanism, the model doesn't need to compress the information of all the tags into a single vector. It can attend to a specific tag based on the decoder state.

5.4 Hierarchical Attention

We investigate if using Hierarchical Attention (Libovický and Helcl, 2017) instead of just concatenating the two context vectors for lemma and the tags as done in Equation 8 proves advantageous. Libovický and Helcl (2017) proposed Hierarchical Attention technique for combining the context vectors in case of multiple source sequences.

$$a_h = \sigma(v \tanh(w_{h_l^*} h_l^* + w_{h_{tg}^*} h_{tg}^* + w_s s_t + b)) \tag{16}$$

$$h_t^* = a_h h_{l_t}^* + (1 - a_h) h_{tg_t}^* \tag{17}$$

After computing the individual context vectors, a scalar $a_h \in [0, 1]$ is calculated. This scalar corresponds to how the attention should be divided between the lemma and the tag. The combined context vector is obtained by taking the weighted average, as shown above.

Compared to the concatenating the context vectors (as done in our submission), using hierarchical attention gives worse results (46.60% average accuracy as compared to 49.08% average accuracy on development set for low resource setting). This is possibly because of the increase in number of parameters to learn and the additional non linearities such as sigmoid and $\tanh$ which lead to vanishing gradient problem.

6 Conclusion

In this paper, we described IITBHU–IIITH system for Task 1 of CoNLL–SIGMORPHON 2018 Shared Task. Our system is one of the top performing systems in this edition of the shared task and beats the baseline by large margins. Even though our approach was completely based on neural networks, our system works very well for low resource setting.

We conclude that neural network architectures with explicit copying mechanism (like Pointer-Generator Network) perform well in Morphological Inflection task even on low resource setting.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2004–2015.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1024–1029.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 31–39, Vancouver. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

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*, Brussels, Belgium. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 1185–1195.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 634–643.

Mans Hulden, Markus Forsberg, and Malin Ahlberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 569–578.

Katharina Kann and Hinrich Schütze. 2016. MED: the LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, Berlin, Germany, August 11, 2016*, pages 62–70.

Katharina Kann and Hinrich Schütze. 2017. The lmu system for the conll-sigmorphon 2017 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 40–48, Vancouver. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jindrich Libovický and Jindrich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 196–202.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection, Vancouver, BC, Canada, August 3-4, 2017*, pages 49–57.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 922–931.

Garrett Nicolai, Bradley Hauer, Mohammad Motallebi, Saeed Najafi, and Grzegorz Kondrak. 2017. If you can't beat them, join them: the university of alberta system description. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 79–84, Vancouver. Association for Computational Linguistics.

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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99, Vancouver. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. 2017. Deep sets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3394–3404.

Chunting Zhou and Graham Neubig. 2017. Morphological inflection generation with multi-space variational encoder-decoders. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 58–65, Vancouver. Association for Computational Linguistics.