Morphological Reinflection in Context: CU Boulder's Submission to CoNLL-SIGMORPHON 2018 Shared Task

Ling Liu and **Ilamvazhuthy Subbiah** and **Adam Wiemerslage** and **Jonathan Lilley** and **Sarah R. Moeller**
University of Colorado
first.last@colorado.edu

Abstract

This paper describes two systems for the second subtask of CoNLL-SIGMORPHON 2018 shared task on universal morphological reinflection submitted by the University of Colorado Boulder team. Both systems are implementations of RNN encoder-decoder models with soft attention. The first system is similar to the baseline system with minor differences in architecture and parameters, and is implemented using PyTorch. It works for both track 1 and track 2 of the subtask and generally outperforms the baseline at low data settings in both tracks. The second system predicts the morphosyntactic description (MSD) of the lemma to be inflected using an MSD prediction model. The data for subtask 2 is processed and reformatted to subtask 1 data format to train an inflection model. Then the inflection model predicts the inflected form for the target lemma given the predicted MSD. This system achieves higher accuracies than the first system when the training data is the most limited, though it does not perform better when the training data is abundant.

1 Introduction

Several natural language processing tasks can benefit from representational power in a computational model at the level of morphology. The task of morphological inflection has been explored recently in great depth (Cotterell et al., 2016, 2017), resulting in several effective models for that task. Of particular note is an architecture proposed by Kann and Schütze (2016), which is modeled after an encoder-decoder model that found success in machine translation (Cho et al., 2014).

A related, but relatively unexplored task is that of morphological inflection in context. This paper documents the University of Colorado Boulder's system for that task (subtask2) in the CoNLL-SIGMORPHON 2018 shared task. We experimented with a model very similar to the provided baseline, which computes the context for a given inflection as the concatenation of word and MSD embeddings to the left and right of the word that is to be inflected. During encoding of an input sequence, the context vector is concatenated with the character embedding at each time step.

We also experimented with an encoder comprising of three separate LSTMs whose output states are concatenated and used to predict the MSD for the lemma to be inflected. We then use a second encoder-decoder network to perform inflection over the given lemma according to that MSD, thus formulating the second portion into the problem in task1 where the training data are pairs of lemma, inflected word form, and the MSD for the inflection and the task is to predict the inflected word form given the lemma and MSD.

We find that the first system described here outperforms the second one when there is ample training data, whereas the latter performs better when the training data is scarce.

2 Task and data description

The shared task is broken into 2 subtasks. This paper presents systems that participated only in subtask2. There are seven languages for this task: German, English, Spanish, Finnish, French, Russian, and Swedish; and 3 data settings: low (< 100 sentence examples), medium (< 900 sentence examples), and high (< 8000 sentence examples). Each data setting varies across languages with regard to the number of training sentence examples.

Within subtask 2 there are two tracks. Both tracks present each problem in context, that is, given some lemma and the word forms surrounding it in a sentence, the goal is to generate the correctly inflected form of that lemma. In track one, the MSD and lemma for each word in the sentence

Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection, pages 86–92,
Brussels, Belgium, October 31, 2018.

is available, whereas in track 2 only the inflected word form is available. For testing in both tracks, only the lemma of the form that should be inflected is provided. More details about the tasks and data can be found in Cotterell et al. (2018).

3 System description

3.1 System 1

Our first system is similar to the subtask 2 baseline system provided by the shared task organizers (Cotterell et al., 2018) but with a few changes in the architecture and parameters. It is an encoder-decoder model with soft attention (Bahdanau et al., 2015) implemented with PyTorch based on the PyTorch tutorial of translation with a sequence to sequence network and attention,[1] and it works for both track 1 and track 2.

Architecture The encoder is a single layer Gated Recurrent Unit (GRU) (Cho et al., 2014). It takes as input the concatenation of the context embedding and the embedded characters in the lemma, and outputs a sequence of state vectors, which are then translated into a sequence of embeddings by a one-layer GRU decoder using an attention mechanism. The embeddings are then transformed into output characters by a log softmax layer. For track 1, the context embedding is the concatenation of word embeddings for the previous inflected word form, previous lemma, previous MSD, current lemma, next inflected word form, next lemma, and next MSD. If a word is at the beginning of the sentence, we add a special symbol $\langle SOS \rangle$ as its history context, and if a word is at the end of the sentence, we add another special symbol $\langle EOS \rangle$ as its future context. The context embedding for track 1 is illustrated in the bottom part of Figure 1. For track 2, the context embedding is the concatenation of word embeddings for the previous inflected word form, current lemma, and next inflected word form as is shown in Figure 2. Special symbols indicating the beginning and end of sentences are also used. For the character embedding of lemmas, we also used $\langle SOS \rangle$ and $\langle EOS \rangle$ to indicate the beginning and end of the lemma.

The decoder starts decoding with input as $\langle SOS \rangle$ and hidden state as the last hidden state of the encoder. An attention mechanism is implemented to

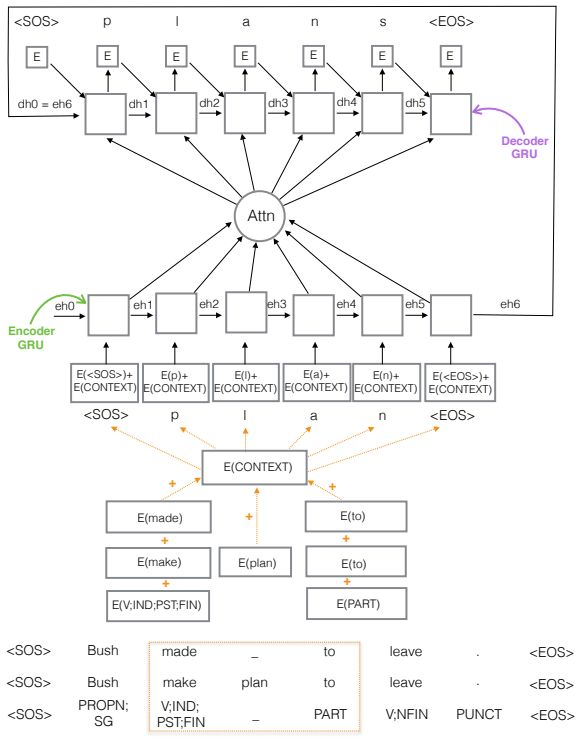

Figure 1: Architecture of system 1 with track 1 context embedding

allow the decoder network to focus on different parts of the encoder's outputs at each step of generation. It is implemented as another feed-forward layer which takes as its input the decoder's input and hidden states and calculates a set of attention weights. We multiply the attention weights with the encoder output vectors to create a weighted combination. This weighted combination will go through a non-linear ReLU layer before going to the GRU process. In addition, a dropout layer is added to the decoder input to deal with overfitting. The dropout rate is 0.1. The decoding process stops when the end symbol $\langle EOS \rangle$ is generated. It may also stop early when a maximum prediction length has been reached. The maximum prediction length is set at 50. The overall architecture of this system as to track 1 is shown in Figure 1. For track 2, only the context embedding is different, i.e. the context is the concatenation of the previous inflected word form, the target lemma and next inflected word form embeddings.

Data To train the model for track 1, the model is trained to make predictions for only entries whose part-of-speech (POS) are verbs, nouns, or adjectives. For track 2, the model is trained to make

[1] https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

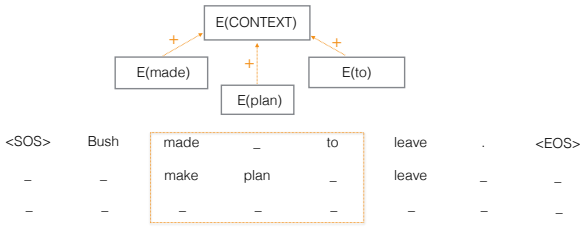

Figure 2: Context embedding for system 1 at track 2

predictions for entries where the lemma form is provided.

Settings and Hyper-parameters The model is trained to minimize the negative log likelihood loss (NLLLoss) on the training data. The optimization method is the Adam algorithm (Kingma and Ba, 2015) with a learning rate of 0.00005. As to other hyper-parameters, we use an embedding size of 100 for the character, lemma, word-form and MSD embeddings. The hidden size is 800 for track 1 and 400 for track 2.

3.2 System 2: MSD prediction and inflection

In our second approach we reformulate the task 2 problem as a task 1 problem. This approach involves predicting the morphosyntactic descriptions of the lemma in question, given the inflected word forms, the lemmas and the MSDs for the rest of the sentence. Once we have the predicted MSDs, we use a task 1 inflection model to get the inflected form. This section describes the architecture for the MSD prediction model followed by the inflection model. This system is only for track 1 as it relies on the morphosyntactic descriptions.

3.2.1 MSD prediction model

The MSD prediction model uses a many-to-one encoder-decoder neural network to predict the MSDs of the lemma to be inflected.

Architecture The encoder uses three separate single-layer bidirectional LSTMs ($LSTM_{left}$, $LSTM_{base}$ and $LSTM_{right}$) to encode the input into a fixed length vector c. It is based on Vylomova et al. (2017). $LSTM_{left}$ takes as input the sequence that is to the left of the current lemma in the sentence and computes the hidden states $h_l^{(t)} \in \mathbb{R}^H$.

$$h_l^{(t)} = f_l(x^{(t)}, h_l^{(t-1)})$$

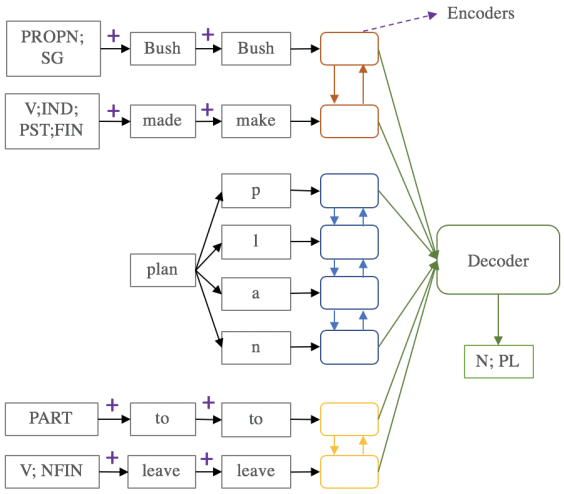

Figure 3: Architecture of the MSD prediction model

where $x^{(t)} \in \mathbb{R}^{3E}$ is a concatenation of the inflected word form embedding $e(w_t)$, the lemma embedding $e(l_t)$ and the MSD embedding $e(msd_t)$ for some word t in the sentence; $e(w_t), e(l_t), e(msd_t) \in \mathbb{R}^E$, where E is the dimension of the embedding layer.

In a similar fashion, $LSTM_{right}$ takes as input the sequence that is to the right of the current lemma in the sentence.

$$h_r^{(t)} = f_r(x^{(t)}, h_r^{(t-1)})$$

$LSTM_{base}$ takes as input the sequence of character embeddings, $e(c_t)$, concatenated with the lemma embedding, $e(l)$, for every character c in the lemma.

$$h_b^{(t)} = f_b(x_b^{(t)}, h_b^{(t-1)})$$

The input to the decoder is the vector $c \in \mathbb{R}^{3H}$ which is the concatenation of the final states from the three encoder LSTMs.

$$c = [h_l^{(T_l)}; h_r^{(T_r)}; h_b^{(T_b)}]$$

The decoder acts as a classifier that classifies the input into one of the possible MSD combinations.

$$h_d = f_d(c, h_d^{(0)})$$

$$P_{msd} = softmax(W.h_d + b)$$

The high level architecture of the model is shown in Figure 3.

Data To train the MSD prediction model,

we used only lemmas whose POS are verbs, nouns or adjectives.

Settings and Hyper-parameters We used a hidden size of 100 and an embedding size of 100 for the character, lemma, word and MSD embeddings. We used Adam as the optimizer with a learning rate of 0.0005.

3.2.2 Inflection model

The Inflection model outputs the inflected word-form given a lemma and its associated morphosyntactic features (for example, *touch + V;V.PTCP;PRS* $\Rightarrow$ *touching*). It is an encoder-decoder soft-attention based neural network that takes as input the sequence of lemma characters and the morphosyntactic descriptions, and produces a sequence of characters as output. The Inflection model is based on University of Colorado Boulder's submission to CoNLL-SIGMORPHON 2017 Shared Task (Silfverberg et al., 2017).

Architecture The encoder is a single layer bi-directional GRU that takes as input the embeddings $e(.)$ of lemma characters and the morphosyntactic descriptions, and produces a sequence of state vectors. The decoder then uses this sequence of state vectors to generate the sequence of output embeddings. At each stage in the decoding process, the decoder uses the following to compute the current state vector:

- the previous decoder hidden state.
- the previous output embedding.
- A weighted sum of all the encoder states.

The decoding process starts with the embedding for the word boundary symbol $\langle EOS \rangle$ and a randomly initialized hidden state h_0. The weights for the encoder states is computed using an Attention mechanism which uses the previous decoder state as input. The weights are normalized using a softmax function. The overall architecture of the inflection model is shown in Figure 4.

Data The data to train the inflection model was generated from the task2 training data by taking out entries of verbs, nouns or adjectives and putting them into task 1 data format. The amount of training data we get in this way is comparable

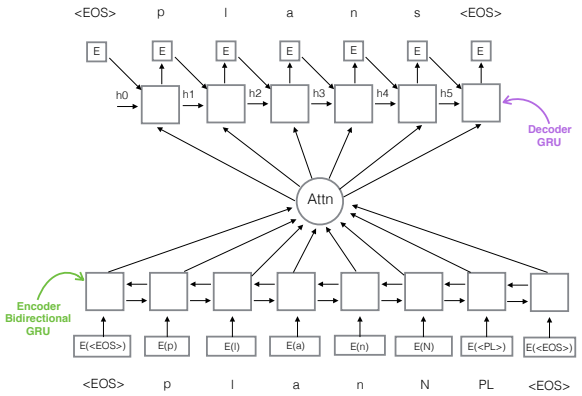

Figure 4: Architecture of the inflection model

to the data size of task 1 at different data settings for all the 7 languages; more than 10000 examples for the high data setting, more than 2000 examples for the medium data setting, and more than 250 examples for the low data setting.

Settings and Hyper-parameters The inflection model uses an embedding size of 100 and a hidden state of size 100 for the encoder and the decoder. The data is processed in batches of 20. Masking is used to mask part of the input sequences which are shorter than the maximum length in the batch. Stochastic Gradient Descent (SGD) with gradient clipping is used for optimization and the loss function is NLLLoss.

4 Experiments

4.1 Exploratory experiment: lemma copying

As the first exploration of the task and an evaluation of task complexity, we experimented by copying the lemma directly. In other words, we simply guess that the inflected form of a lemma in the context is the lemma itself. This experiment will be referred to as the copy system going further.

4.2 System 1

We tuned the architecture and the parameters for the first system on track 1 and finally settled on the architecture and parameters described in section 3.1 for track 1 and track 2. For both tracks, we train the model for 50 epochs at the low data setting and 40 epochs at medium and high data settings, and use the model at the epoch which gets the highest accuracy on the development set to make predictions on the test set.

	HIGH				MEDIUM				LOW			
LANGUAGE	COPY	SYS 1	SYS 2	BASELINE	COPY	SYS 1	SYS 2	BASELINE	COPY	SYS 1	SYS 2	BASELINE
DE	58.95	63.3	62.49	**64.51**	**58.95**	55.11	55.11	54.4	**58.95**	10.41	*32.15*	0.2
EN	62.64	**76.23**	68.58	72.91	62.64	65.16	**66.57**	60.02	**62.64**	57.6	*59.62*	1.81
ES	25.53	51.75	36.75	**53.44**	25.53	**41.8**	32.68	23.14	25.53	25.53	**27.77**	8.98
FI	22.36	43.07	30.24	**49.05**	22.36	23.38	23	**28.21**	**22.36**	7.12	*11.44*	0.76
FR	23.63	58.04	60.9	**63.54**	23.63	44.2	42.16	**45.01**	23.63	25.66	**26.88**	0
RU	18.37	65.16	55.52	**71.18**	18.37	44.58	40.46	**50.3**	18.37	15.06	**22.09**	0
SV	32.45	60.53	37.77	**62.23**	32.45	**49.68**	34.57	47.55	**32.45**	20.74	*29.04*	1.17
AVERAGE	34.85	59.73	50.32	**62.41**	34.85	**46.27**	42.08	44.09	**34.85**	23.16	*29.86*	1.85

Table 1: Track 1 accuracies for original form of the copy system, system 1, system2, and baseline

	HIGH				MEDIUM				LOW			
LANGUAGE	COPY	SYS 1	SYS 2	BASELINE	COPY	SYS 1	SYS 2	BASELINE	COPY	SYS 1	SYS 2	BASELINE
DE	58.95	59.96	-	**65.72**	**58.95**	49.54	-	56.93	**58.95**	11.02	-	0.1
EN	62.64	**70.8**	-	70.39	**62.64**	61.53	-	57.6	**62.64**	58.91	-	2.22
ES	25.53	45.86	-	**51.05**	25.53	35.34	-	**41.23**	25.53	**27.91**	-	8.98
FI	22.36	24.02	-	**34.82**	**22.36**	17.15	-	19.19	**22.36**	7.88	-	0.38
FR	23.63	48.27	-	**58.45**	23.63	**36.05**	-	21.38	**23.63**	23.01	-	0
RU	18.37	39.76	-	**46.89**	18.37	27.21	-	**30.52**	18.37	**21.08**	-	2.71
SV	32.45	**54.15**	-	54.04	32.45	41.17	-	**43.09**	**32.45**	16.49	-	0.96
AVERAGE	34.85	48.97	-	**54.48**	34.85	38.29	-	**38.56**	**34.85**	23.76	-	2.19

Table 2: Track 2 accuracies for original form of the copy system, system 1, system2, and baseline

4.3 System 2

In our experiments for System 2, we train the MSD prediction model for 10 epochs on all data settings. To train the inflection model we used 200 epochs for the low data setting and 100 epochs for the medium and high data settings. We also experimented with higher embedding and hidden sizes of 200 for the MSD prediction model and we found no significant improvements.

5 Results and discussion

The evaluation result as to original forms for track 1 is shown in Table 1. In general, our first system (SYS 1) outperforms the second system (SYS 2) in both high and medium data settings, though neither of them get a higher averaged accuracy than the baseline system when the training data size is high and the first system is only marginally better than the baseline when the training data is of medium size. For the four systems summarized in the table, our first system performs the best only with English at high data setting, and it achieves the highest accuracies with Spanish and Swedish at medium data setting. The second system outperforms the other three systems with English at medium data setting. However, when the training data is the most limited, i.e. at the low data setting, the second system outperforms both the first system and the baseline as to average accuracy over the seven languages, though it is still worse than the copy system. To be specific, direct lemma copy produces the best results for German, English, Finnish and Swedish among the four systems at the low data setting. The second system outperforms the other three systems with Spanish, French, and Russian.

Table 2 provides the evaluation results for the baseline system, our first system and the copy system, as to original forms for track 2. Our second system relies on the prediction of MSDs and does not work for track 2. For this track, we see the pattern of the system performance is similar to that of track 1. That is, the baseline system generally outperforms our first system and the copy system at the high data setting and gets very close to the first system in terms of averaged accuracies at the medium data setting, and our first system outperforms the baseline by a large margin at the low data setting though it is still worse than the copy system.

When the training data is the most limited, comparing the results of system 1 for track 1 and track 2, shows that track 1 results are not better than track 2 results, indicating that the MSD and lemma information does not really help with the performance of the first system when a limited amount of data is available. However, the second system outperforms the first system on track 1 for all languages, and the results of SYS 2 on track 1 are

higher than the results of SYS 1 on track 2 on most languages (except Spanish, for which SYS 2 track 1 is a mere 0.14% lower). This suggests that, when training data is very limited, the MSDs introduce a lot of ambiguity if only used as contextual information. On the other hand, if we first predict the MSD as our second system does, the ambiguity is reduced and thus the system generates better predictions. German is the language where the second system is most significantly better than the first system on the track 1 low data setting. German is a language with much ambiguity in its inflected forms. For a German word form, there can be as many as 40 different readings (Müller and Schütze, 2015). This fact also supports the ambiguity explanation for the difference in the performance of our first and second systems.

In the low setting, Finnish, German and Russian have the lowest scores for our first model. Finnish and Russian are the two languages with the most complex inflection systems in the sense that they have the highest number of distinct MSDs. In the high data setting, counting only the parts-of-speech the model is supposed to predict, i.e. nouns, verbs and adjectives, there are 346 distinct MSDs in Finnish training data and 345 in Russian training data. The rich inflection requires more data for the model to learn. Though German has less distinct MSDs than Finnish and Russian, its inflection is also complex in the sense that it's less predictable: Unlike Russian and Finnish which use almost exclusively suffixes, German is not primarily suffixing but employs prefixing, circumfixing, and umlauting. Its inflectional rules are less regular than Finnish or Russian. A qualitative analysis of the predictions in the low data setting, finds that our first model tends to make changes in the stem for German though not for Finnish and Russian, and the wrong changes in the stem cause wrong predictions for German words while for Finnish and Russian the errors are mainly wrong suffixation. This agrees with the distinct features of their inflection systems.

The intuition behind just copying the lemma when the training data is limited, is the linguistic observation that the lemma form is usually the most frequently used form and thus any uninformative inflection tends to be less likely than the lemma.

6 Conclusion

For this task, we explored the performance of RNN encoder-decoder models with soft attention as to predicting the inflected forms of a lemma in context. We developed two systems by implementing the encoder-decoder model in different ways. We found that when the training data is very limited, morpho-syntactic descriptions contribute to better prediction results. Though both of our systems outperform the baseline at the low data setting, none of the systems are better than the blind guess of the inflected form being the lemma itself. However, when the training data is abundant, neural network systems outperform the lemma copying approach.

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*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, Vancouver, Canada. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task: Morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Berlin, Germany. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*.

Thomas Müller and Hinrich Schütze. 2015. Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 526–536.

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

Ekaterina Vylomova, Ryan Cotterell, Timothy Baldwin, and Trevor Cohn. 2017. Context-aware prediction of derivational word-forms. *CoRR*, abs/1702.06675.