Morphological reinflection with convolutional neural networks

Robert Östling
Department of Modern Languages, University of Helsinki
PL 24 (Unionsgatan 40, A316)
00014 Helsingfors universitet, Finland
robert.ostling@helsinki.fi

Abstract

We present a system for morphological reinflection based on an encoder-decoder neural network model with extra convolutional layers. In spite of its simplicity, the method performs reasonably well on all the languages of the SIGMORPHON 2016 shared task, particularly for the most challenging problem of limited-resources reinflection (track 2, task 3). We also find that using *only* convolution achieves surprisingly good results in this task, surpassing the accuracy of our encoder-decoder model for several languages.

1 Introduction

Morphological reinflection is the task of predicting one form from a morphological paradigm given another form, e.g. predicting the English present participle *ringing* given the past tense *rang*. The SIGMORPHON shared task considers three variants of this problem, with decreasing amounts of information available beyond the source form and the morphological features of the target form:

1. The source form is always the citation form.
2. The source form's morphological features are not fixed, but given.
3. Only the source form itself is given.

The first and simplest case is the most well-researched, and is essentially equivalent to the task of predicting morphological paradigms.

This paper presents our system for morphological reinflection, which was submitted for the SIGMORPHON 2016 shared task. To complement the description given here, the source code of our implementation is available as free software.[1]

[1]https://github.com/robertostling/sigmorphon2016-system

2 Background

In general, morphological reinflection can be solved by applying any technique for morphological analysis followed by morphological generation. These tasks have traditionally been performed using manually specified rules, a slow and expensive process. Recently, there has been an increased interest in methods for learning morphological transformations automatically from data, which is also the setting of the SIGMORPHON 2016 shared task.

This work is based on that of Faruqui et al. (2016), who use a sequence-to-sequence model similar to that commonly used in machine translation (Sutskever et al., 2014). Their method is very simple: for each language and morphological feature set, they train a separate model with a character-level bidirectional LSTM encoder (where only the final hidden states are used), and an LSTM decoder whose inputs are the encoded input as well as the input character sequence.

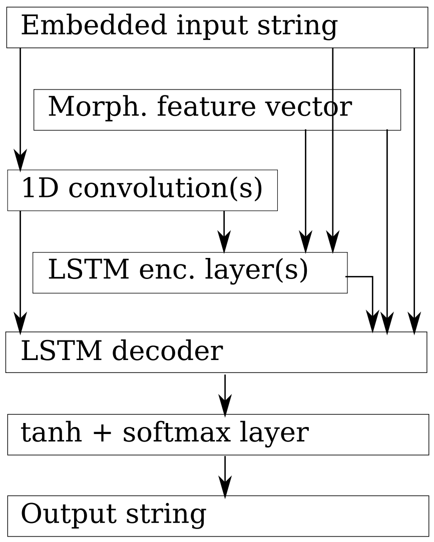
3 Model

We propose modifying the model of Faruqui et al. (2016) by:

1. using a single decoder, rather than one for each combination of morphological features (which could lead to data sparsity for languages with complex morphology and large paradigms),
2. using both the raw letter sequence of the source string and its convolution as inputs,
3. using deeper LSTM units for the decoder.

Although this model was originally designed for inflection generation given a lemma, it can trivially be used for reinflection by using inflected forms rather than lemmas as input. Thus, we use exactly the same model for the first and third task,

Figure 1: Structure of our convolutional encoder-decoder model (note that convolutional layers are not present in all configurations).

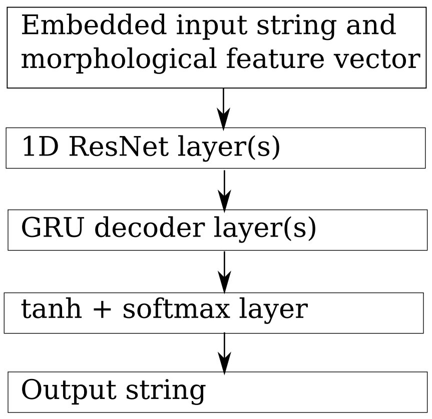


and for the second task where morphological features are given for the source form, we include those features along with the target form features (which are given in all three tasks).

In our experiments, we use 4 convolutional layers and 2 stacked LSTMs (Hochreiter and Schmidhuber, 1997). We use 256 LSTM units (for both the encoder and decoder), 64-dimensional character embeddings and 64 convolutional filters of width 3 for each layer. The LSTM outputs were projected through a fully connected hidden layer with 64 units, and finally through a fully connected layer with softmax activations over the alphabet of the language in question. Morphological features are encoded as binary vectors, which are concatenated with the character embeddings (and, when used, convolved character embeddings) to form the input of the decoder. We then used the Adam algorithm (Kingma and Ba, 2014) for optimization, where the training objective is the cross-entropy of the target strings. For decoding, we use beam search with a beam size of 4. The model architecture is summarized in figure 1.

To further explore the effect of using convolutional layers in isolation, we also performed follow-up experiments after the shared task submission. In these experiments we used an even simpler architecture without any encoder, instead we used a 1-dimensional residual network architecture (He et al., 2016, figure 1b) with constant

Figure 2: Structure of our purely convolutional model (note that GRU layers are not present in all configurations).



size across layers, followed by either one or zero Gated Recurrent Unit layers (Cho et al., 2014). The output vector of each residual layer (which contains two convolutional layers with Batch Normalization (Ioffe and Szegedy, 2015) and rectified linear units after each) is combined with the vector of the previous layer by addition, which means that the output is the sum of the input and the output of each layer. This direct additive coupling between layers at different depth allows very deep networks to be trained efficiently. In this work we use up to 12 residual layers, corresponding to a total of 24 convolutional layers.

In these experiments (unlike the encoder-decoder model), dropout (Srivastava et al., 2014) was used for regularization, with a dropout factor of 50%. The morphological features of the target form are concatenated to the 128-dimensional character embeddings at the top convolutional layer, so the total number of filters for each layer is $128 + n$ in order to keep the architecture simple and uniform, where n is the number of different morphological features in the given language. Decoding is done by choosing the single most probable symbol at each letter position, according to the final softmax layer. This model is summarized in figure 2.

4 Evaluation

All results reported in this section refer to accuracy, computed using the official SIGMORPHON 2016 development data and scoring script. Table 1 on the following page shows the result on the official test set, and a full comparison to other systems

Table 1: Results of our convolutional encoder-decoder system on the official SIGMORPHON shared task test set.

Language	Accuracy (percent)		
	Task 1	Task 2	Task 3
Arabic	89.52	69.53	70.43
Finnish	95.14	88.42	87.55
Georgian	97.02	92.84	91.85
German	94.40	91.73	89.14
Hungarian	98.38	96.25	96.46
Maltese	86.16	73.17	75.54
Navajo	82.10	77.37	83.21
Russian	89.94	86.60	84.59
Spanish	98.35	95.35	94.85
Turkish	97.93	91.69	91.25

is available on the shared task website[2] (our system is labeled 'HEL').

We participate only in track 2, which only allows training data from the same task that is evaluated. Training data from other (lower-numbered) tasks, as track 1 allows, could trivially be appended to the training data of our model, but this was not done since we focused on exploring the core problem of learning reinflection. The same constraints are followed in all experiments described here.

Note that due to time constraints, we were not able to explore the full set of parameters before submitting the test set results. Of the models that had finished training by the deadline, we chose the one which had the highest accuracy on the development set. The results reported here are from later experiments which were carried out to systematically test the effects of our proposed changes. Table 2 shows that using convolutional layers improves accuracy in almost all cases, whereas adding an extra LSTM layer does not bring any systematic improvement.

Results when using only convolutional layers or convolutional layers followed by a GRU recurrent layer can be found in table 3 on the following page. To our surprise, we found that convolution alone is sufficient to achieve results comparable to or better than several of the other systems in the shared task, and for some languages it beats our own submitted results. There is no clear benefit across languages of adding a final GRU decoder layer, but increasing the depth of the network and in particular the width of the convolution seem to benefit accuracy.

Table 2: Results of our convolutional encoder-decoder system on the official SIGMORPHON shared task development set for task 3 (reinflection). The first column contains results of models with both convolutions (4 layers) and deep LSTMs (2 layers), the second uses a single LSTM layer, and the third one uses no convolutional layers.

Language	Accuracy (percent)		
	both	-deep	-conv
Arabic	66.9	70.8	**75.8**
Finnish	85.5	**88.4**	80.9
Georgian	**92.3**	91.9	87.1
German	**89.6**	87.2	88.7
Hungarian	**97.1**	94.0	95.6
Maltese	**76.1**	74.0	74.9
Navajo	**89.6**	87.2	85.1
Russian	83.2	**84.1**	82.2
Spanish	93.6	**94.3**	91.1
Turkish	**89.7**	88.8	80.4

5 Conclusions

We find that the model of Faruqui et al. (2016) can be extended to the task of reinflection and delivers very good levels of accuracy across languages, and that adding convolutional layers consistently improves accuracy.

Further experiments show, to our surprise, that a simple and purely convolutional architecture designed for image classification in many cases achieves an even higher accuracy. Although convolutional architectures have become standard (along with recurrent neural networks) in many text *encoding* tasks, this is one of rather few examples of where they have been successfully used for text *generation*.

Acknowledgments

This work was carried out using the computing resources of CSC.[3]

[2] `http://ryancotterell.github.io/sigmorphon2016/`

[3] `https://www.csc.fi/`

Table 3: Results of our purely convolution system (not submitted) on the official SIGMORPHON shared task development set for task 3 (reinflection). System configurations are given on the form “convolutional layers–filter size”.

Language	Accuracy (percent)									
	With GRU decoder					Without GRU decoder				
	24–7	16–7	8–7	24–5	24–3	24–7	16–7	8–7	24–5	24–3
Arabic	**74.2**	69.6	63.7	71.8	47.7	71.9	68.1	67.4	65.0	55.5
Finnish	89.6	90.9	84.9	85.5	90.4	**91.3**	89.2	91.0	88.8	86.9
Georgian	91.2	91.4	91.3	**91.5**	90.1	89.9	89.7	90.3	91.0	89.6
German	89.0	89.8	89.1	89.6	88.8	88.9	**89.9**	89.6	89.8	88.9
Hungarian	93.5	**96.0**	89.8	93.8	92.0	92.2	90.0	88.0	90.9	90.0
Maltese	63.0	63.2	60.4	50.1	63.2	**66.0**	61.1	54.1	61.2	64.6
Navajo	78.8	81.9	72.4	78.8	50.0	**84.3**	80.5	49.6	68.5	31.4
Russian	84.8	85.0	**86.1**	85.4	83.2	85.4	86.0	85.0	86.1	82.2
Spanish	**95.5**	92.6	94.5	95.3	92.7	94.7	94.8	94.9	94.1	95.2
Turkish	91.4	91.4	92.1	90.8	89.7	**92.8**	91.1	90.7	90.7	90.4

References

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, 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)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proc. of NAACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings.

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

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.