Character Sequence-to-Sequence Model with Global Attention for Universal Morphological Reinflection

Qile Zhu, Yanjun Li, Xiaolin Li
Large-scale Intelligent Systems Laboratory
NSF Center for Big Learning
University of Florida
{valder, yanjun.li}@ufl.edu, andyli@ece.ufl.edu

Abstract

This paper presents a neural network based approach for the CoNLL-SIGMORPHON-2017 Shared Task 1 on morphological reinflection. We propose an encoder-decoder architecture to model this morphological reinflection problem. For an input word, every character is encoded through a Bi-directional Gated Recurrent Unit (GRU) network. Another GRU network is deployed as a decoder to generate the inflection. We participate in Task 1, which includes 52 languages without using external resources. In each language, three training sets are provided (high, medium, and low respectively represent the amount of training data; Scottish Gaelic only has medium and low), totally 155 training sets. Due to time constraints, we only search for optimized parameters of our model based on the Albanian dataset. The source code of our model is available at https://github.com/valdersoul/conll2017.

1 Introduction

A linguistic paradigm is the complete set of related word forms associated with a given lexeme. Within the paradigm, inflected word forms of lexemes are defined by the requirements of syntactic rules. A word's form reflects syntactic and semantic features that are expressed by the word, such as the conjugations of verbs, and the declensions of nouns (Cotterell et al., 2017). For example, every English count noun has both singular and plural forms, known as the inflected forms of the noun. Different languages have various degrees of inflection. Some can be highly inflected, such as Latin, Greek, Spanish, Biblical Hebrew, and Sanskrit, and some can be weakly inflected, such as English. An example is shown in table 1.

	inflection	tags
release	release	V;NFIN
release	releases	V;3;SG;PRS
release	releasing	V;V.PTCP;PRS
release	released	V;PST
release	released	V;V.PTCP;PST

Table 1: An example of an inflection table from word "release".

The issue of analyzing and generating different morphological forms has received considerable critical attention. Errors in the understanding of morphological forms can seriously harm performance in the machine translation and question answering systems. On the other hand, applying inflection generation as a post-processing step has been shown to be beneficial to reducing the data sparsity when translating morphologically-rich languages (Minkov et al., 2007).

For the CoNLL-SIGMORPHON-2017 Shared Task 1 (Cotterell et al., 2017) on morphological reinflection, given a lemma (the dictionary form of a word) and target morphosyntactic descriptions, a target inflected form is required to be generated across 52 different languages. In each of these languages, there are three training sets (high, medium, and low) representing different amount of training data (Scottish Gaelic only has medium and low).

2 Related Work

Inflection generation can be modeled as string transduction and consists of three major components: (1) Aligning characters forms; (2) Extracting string transformation rules; (3) Applying rules to new root forms (Faruqui et al., 2016).

Recently, end-to-end deep learning approaches achieve state-of-the-art performance across many different datasets. LMU system ranked first in SIGMORPHON shared task (Kann and Schütze,

Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection, pages 85–89,
Vancouver, Canada, August 3–4, 2017.

2016). It used an encoder-decoder structure with attention mechanism to do translation from root word to its inflection. At the same time, convolutional neural networks have been leveraged to extract features from root words (Ostling, 2016). Faruqui et al. (2016) added language model interpolation into the encoder-decoder structure and trained the neural network in both supervised and semi-supervised settings, and achieved state-of-the-art performance in Spanish verb and Finnish noun and adjective datasets.

Our system leverages a sequence-to-sequence model similar to Faruqui et al. (2016). For each language and training set, we train a separate model using a character-level bidirectional GRU encoder and a single layer GRU decoder with a global attention model (Luong et al., 2015).

3 Model

Our system for this Shared Task 1 is based on an encoder-decoder model proposed by Bahdanau et al. (2014) for neural machine translation. The RNN unit we use in our system is GRU (Cho et al., 2014). Fig. 1 shows our overall architecture.

The GRU reads an input sequence and encodes each input as a fixed length vector h_i, which is computed by

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \quad (1)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \quad (2)$$

$$\hat{h}_t = tanh(r_t \circ U h_{t-1} + W_h x_t) \quad (3)$$

$$h_t = (1 - z_t) \circ \hat{h}_t + z_t \circ h_{t-1} \quad (4)$$

To obtain global information of each input, we use a bidirectional GRU and concatenate each hidden state as one vector, $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ to be the output of encoder's hidden state. For the decoder, we use a single layer GRU. Our model has two input streams, one is the characters and the other is the morphological tags. We only encode the input characters, and make the morphological tags as another input feature to contribute to the outputs. We pad the morphological tagging sequences to the length of the longest tags in the training sets, and feed them into a fully connected network to produce the feature.

In neural machine translation, the input and output sequences are semantically equivalent. However, morphological inflection of a word has different semantics (Faruqui et al., 2016). So we make the encoded input sequence as a part of the input of decoder together with the morphological tags (including part-of-speech; POS). To get the hidden state of decoder at time step t, we use the previous hidden state h_{t-1}, the decoder input y_{t-1}, the encoder state of the root word e, and the representation of morphological tagging sequence of target form p to compute:

$$h_t = g(h_{t-1}, \{y_{t-1}, e, p\}) \quad (5)$$

where g is the GRU decoder function.

Another difference from machine translation is that our input and output sequence characters may be very similar except the inflections. Take the words *release*, *releasing*, and *released* from English as an example, these three words only differ in the ending characters. To make full use of this similarity, we also add the corresponding input character as a part of the decoder's input, so h_t is computed as

$$h_t = g(h_{t-1}, \{y_{t-1}, x_t, e, p\}) \quad (6)$$

To solve the variable length of input and output sequences, we add paddings as x_t indicating null input.

In the decoding phase, we use a global attention model based on the hidden state of decoder and all the hidden states from the encoder (Luong et al., 2015) to calculate the context vector c_t at time step t as:

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j \quad (7)$$

where α_{tj} is the attention weights, h_j is the output of each hidden state from the encoder. The weights are computed as

$$score_{tj} = tanh((W_a h_t + b_a)^T \cdot h_j) \quad (8)$$

$$\alpha_{tj} = \frac{exp(score_{tj})}{\sum_i exp(score_{ti})} \quad (9)$$

This context vector can be treated as a fixed representation of what has been read from the source for this time step. We concatenate it with the decoder state h_t and feed it through another fully connected network to produce the output distribution (Fig. 2):

$$P_y = softmax(W[c_t; h_t] + b) \quad (10)$$

The loss for time step t is the negative log likelihood of the target w_t:

$$loss_t = -log(P(w_t)) \quad (11)$$

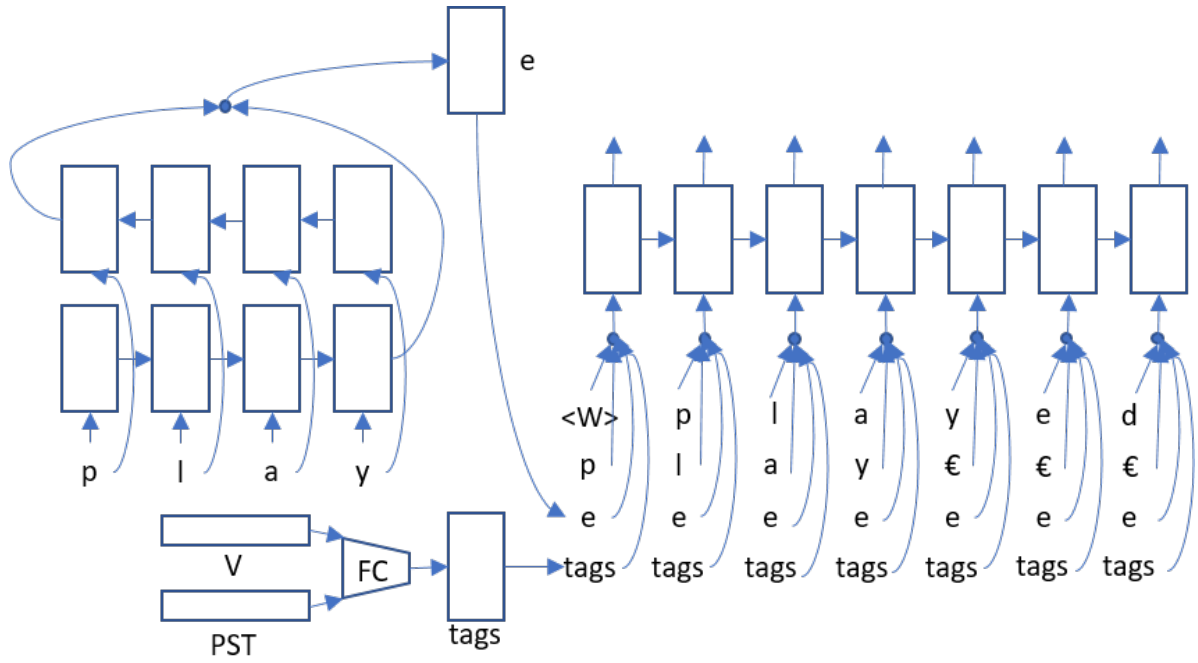

Figure 1: The overall architecture of our approach (without the global attention model and ϵ is the padding character).

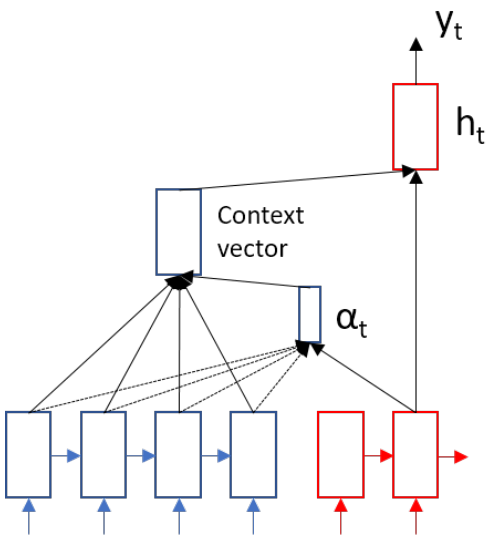

Figure 2: The attention model.

and the overall loss for the whole sequence is computed by:

$$loss = \frac{1}{T}\sum_{t=0}^{T} loss_t \qquad (12)$$

When decoding, we use beam search of size 4 to generate possible output character sequences and rank them by the average probability of characters.

4 Experimental Evaluation

4.1 Data Format

The data provided by Task 1 is the root word and its target morphological tags. We add some special symbols to the character set for every language: "UNK" represents the unknown character, "PAD" is the padding character, "START" denotes the starting of a sequence and "END" represents the ending of a sequence. We only add "START" and "END" to the output sequences. Because the input is fixed, and it is not necessary to make the encoder aware of when the sequence will finish. Although the starting character is not considered in the loss, the ending character is taken into account.

4.2 Training Setting

Due to time limits, we only use the Albanian dataset to do parameter searching. As shown in table 2, we leverage three different groups of parameters based on the variety of the training sets (high/medium/low) for Task 1. We use the same embedding size for characters and morphological tags. The length of morphological tags of a training sample differs from each other, so we pad them to the longest one in each training corpus. We also use a dropout layer after the embedding layer to prevent overfitting.

	Embedding Size	Hidden Size
High	100	200
Medium	100	50
Low	50	20

Table 2: The embedding size and hidden size for three different settings.

	Dropout Rate
High	[0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85]
Medium	[0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7]
Low	[0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6]

Table 3: The dropout rates used for three different settings.

For training, we use Adam algorithm (Kingma and Ba, 2014) and set different minibatch sizes according to various training settings (high/medium/low). Compared to high setting, there are much less training samples in the medium and low. Thus it will take more time to converge if we set the minibatch size too large. We also use early stopping (Caruana et al., 2000) based on the performance of development sets.

4.3 Ensemble

We use different dropout rates to train multiple models in the same training set. Table 3 shows our dropout rates for different models. To select the best result, we use the voting strategy from different models and pick up the answer that appears most likely.

4.4 Results

All the results in this section are evaluated in accuracy for different languages, computed over the official test data. Tables 4, 5, and 6 show the results of our model in different settings.

High			
Top 10		Bottom 10	
Urdu	99.4	French	79.1
Hindi	99.1	Hungarian	78.7
Welsh	98	Serbo Croatian	78.6
Quechua	97.1	Icelandic	78
Haida	97	Romanian	77.3
Khaling	96.6	Faroese	77
Persian	96.3	Finnish	74.4
Basque	96	Irish	73.1
Bengali	96	Navajo	68.5
Estonian	94.8	Latin	54.7

Table 4: Top 10 and bottom 10 performance in the high setting of Task 1.

Medium			
Top 10		Bottom 10	
Bengali	93	Icelandic	57.9
Urdu	90.8	Romanian	57.5
Quechua	90.7	Arabic	55.8
English	88.3	Faroese	52
Hindi	86.8	Northern Sami	50.4
Kurmanji	86.3	Lithuanian	49.2
Portuguese	84.3	Finnish	37.3
Turkish	83.2	Irish	35.4
Haida	81	Latin	30.9
Catalan	80.6	Navajo	28.9

Table 5: Top 10 and bottom 10 performance in the medium setting of Task 1.

Low			
Top 10		Bottom 10	
English	74.7	Finnish	7.6
Norwegian Bokmal	73.6	Latin	7.4
Kurmanji	71.1	Haida	7
Danish	59.7	Northern Sami	4.9
Wwedish	51.7	Albanian	4.5
Urdu	46.9	Khaling	3.2
French	46.2	Arabic	1.5
Norwegian Nynorsk	45.8	Basque	1
Portuguese	44.5	Navajo	0.3
Scottish Gaelic	44	Sorani	0.1

Table 6: Top 10 and bottom 10 performance in the low setting of Task 1.

In each training setting (high/medium/low), we use the same parameters for all languages, instead of optimizing parameters based on different language. It means that our model may not be optimal for some languages, which is the reason why the performance differed a lot from each other. The top languages may have some related properties with Albanian. However, languages like French, Romanian and Latin may not be correctly modeled by our model.

In the low setting of Task 1, we only get 100 training samples for each language. Deep learning may easily overfit and can not generate good results when testing. That is why Haida performs well in high and medium settings while staying at the bottom 10 in the low setting.

5 Conclusion

In this paper, we proposed a character sequence-to-sequence model with global attention to do morphological reinflection and achieved good results in some languages. Due to the time constraint, we only searched for the optimized model based on the Albanian dataset, which may not be suitable for other languages. It might be interest-

ing to add some linguistic features to improve the performance and the generalization of our system.

Acknowledgements

The work presented in this paper is supported in part by National Science Foundation (CNS-1624782) and National Institutes of Health (R01GM110240). The content is solely the responsibility of the authors and does not necessarily represent the official views of the granting agencies.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Rich Caruana, Steve Lawrence, and Lee Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *NIPS*. pages 402–408.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Ryan Cotterell, Christo Kirov, John Walther Géraldine Sylak-Glassman, 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*. Association for Computational Linguistics, Vancouver, Canada.

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

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the sigmorphon 2016 shared task on morphological reinflection. *ACL 2016* page 62.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *ACL*. volume 7, pages 128–135.

Robert Ostling. 2016. Morphological reinflection with convolutional neural networks. *ACL 2016* page 23.