# Attention-free encoder decoder for morphological processing

**Stefan Daniel Dumitrescu and Tiberiu Boros**
Research Institute for Artificial Intelligence “Mihai Drăgănescu”
Romanian Academy
13 September, no. 13, Bucharest, Romania
sdumitrescu@racai.ro, tibi@racai.ro

## Abstract

We present RACAI's Entry for the CoNLL-SIGMORPHON 2018 shared task on universal morphological reinflection. The system is based on an attention-free encoder-decoder neural architecture with a bidirectional LSTM for encoding the input sequence and a unidirectional LSTM for decoding and producing the output. Instead of directly applying a sequence-to-sequence model at character-level we use a dynamic algorithm to align the input and output sequences. Based on these alignments we produce a series of special symbols which are similar to those of a finite-state-transducer (FST).

## 1 Introduction

Languages with rich morphology convey morphological attributes such as gender, case, number, obliqueness through character/grapheme variations applied to the dictionary form of the word (lemma). It is often the case where these variations are obtained by suffixing the word rather than altering random characters, but this does not hold for all languages or irregular word forms. Sill, the variations inside the lemma are usually small, requiring the system just to replace an average of 2.3 letters for the irregular word forms.

In our approach we exploit this property and employ an encoder-decoder sequence-to-sequence model that doesn't require an attention mechanism. This mitigates attention issues such as repeating or skipping character sequences and reduces the need for models with high representational capacity.

We exploit the property that alignments between the input and output character sequences are *monotonic*: for example wordform *men* and lemma *man* share two letters (alignments) in the same order, without inversions. The standard attention mechanism is well-suited for machine learning tasks; however, when it comes to monotonic alignments it sometimes fails to achieve satisfactory results, in most cases due to the fact that repeated characters or character sequences in the input sequence confuse the attention mechanism making it generate loops or skip characters.

There are several proposed methods that try to solve this task with attention mechanisms such as guided attention (Tachibana et al., 2017), location-sensitive attention (Chorowski et al., 2015) and other variations. Still, given the particularities of morphological reinflection, we argue that there is no need for an explicit attention mechanism. Instead we train the decoder to focus on a single input symbol at each time-step and “self-attend” by shifting the input cursor with one position at a time. This method, though developed independently, closely resembles that of Makarov et al. (2017).

In our previous experiments we used this architecture to perform lemmatization (the opposite task of morphological reinflection) and we obtained state-of-the-art results.

In what follows, we will present the attention-free encoder-decoder architecture (Section 2), we show our experimental results (Section 3) and finally we draw conclusions (Section 4).

## 2 Attention-free encoder-decoder

The architecture of our neural network is fairly simple. We use an encoder that “sees” the sequence in both directions and a decoder which is conditioned to produce the output sequence using **focused encoder states** (see below for details) concatenated with trainable embeddings computed on morphological attributes.

As mentioned before, the classical attention mechanism is not well suited for tasks where the alignments between the input and output sequences are monotonic. Instead, connectionist temporal classification (CTC) (Graves et al., 2006) provides better results in these cases. However, CTC requires that the number of input time-steps is much higher than the number of output labels. This renders CTC unsuitable for morphological reinflection as the number of output labels is almost always greater than the number of input characters.

Instead, we propose a simpler algorithm that reduces the model complexity and computational load. Our method requires preexisting alignments between the input and output sequences. These alignments are easy to obtain by exploiting a basic property of morphological reinflection, which also holds for lemmatization: **regardless of the language and irregularity of the word-form, the lemma and the inflected word form share many symbols**.

This implies a high likelihood of aligning identical input and output symbols and does not require Expectation Maximization (EM) for computing alignment probabilities. With this in mind, we propose the following algorithm that:

1. Computes an alignment matrix using dynamic programming;
2. Reads the two sequences in reverse and uses the previously computed matrix, favoring diagonal alignments over other alignments;
3. Generates alignment pairs, whenever the input and output symbols are identical.

Figure 1 describes our approach step-by-step.

The algorithm is a slightly modified dynamic algorithm in the sense that (a) it favors diagonal alignments (to cope with repeating consecutive letters) and (b) it only considers an alignment pair $(i, j)$ if the characters from the source ($s$) and destination ($d$) at the two indexes are identical (i.e. $s_i = d_j$).

Next, we use the produced alignments to generate the training data for our attention-free encoder-decoder model. For our algorithm to work, we need the decoder to keep track of the focused-on character in the input sequence. This is achieved by simulating a FST using neural networks. Given the input sequence $s$, the decoder must produce an output sequence $d'$ which is composed of three specialized labels and arbitrary characters in the vocabulary. The output symbols are:

$s$ - input sequence of size $n$
$d$ - output sequence of size $m$

```
a <- zeros(n+1, m+1)
#initialization
for i=(0,n): a[i,0]<-i
for i=(0,m): a[0,i]<-i

for i=(1,n):
        for j=(1,m):
        if s[i-1]==d[j-1]:
                cost<-0
        else:
                cost<-1
                a[i,j]<-cost+
                  min(a[i-1,j-1],
                      a[i-1,j],
                      a[i,j-1])

alignments={}; pi<-n; pj<-m
while i!=1 or j!=1:
        if i==1: j<-j-1
    else if j==1: i<-i-1
    else:
        if a[i-1,j-1]<=a[i-1,j] and
            a[i-1][j-1]<=a[i,j-1]:
                i<-i-1; j<-j-1
        else if a[i-1][j]<a[i][j-1]:
                i<-i-1
        else:
                j<-j-1
    if s[i]==d[j]:
         alignments<-
            alignments+(i-1,j-1)
return alignments
```

Figure 1: Alignment algorithm

- **Special symbol _INC_**: This means that the current focus-index of the input sequence must be incremented by 1;
- **Special Symbol _COPY_**: The character at the current focus-index must be “copied” in order to compose the final sequence;
- **Special Symbol _EOS_**: The output sequence is complete and the algorithm stops;
- **Any arbitrary character in the vocabulary**: This means that the final sequence must be obtained by adding this character.

At runtime we start by setting the focus-index at

0 and the final sequence to the void string (“”) and we follow the instructions of the decoder output in order to construct the final sequence. During training, it is highly important to do sanity checks on the current focus-index to avoid index out-of-bounds exceptions during the first training epochs when the model has not yet converged. Once the loss is small enough, we found that the model rarely generates these exceptions. However, it is still recommended to keep these checks in place.

To obtain the output sequence $d'$ on which we train our network we use a **fixed-oracle algorithm** that is summarized as:

1. Take every symbol in the output sequence and check if it aligns with a symbol in the input sequence (based on the *alignments* produced by the algorithm in Figure 1);
2. If the output symbol does not align with any character, instruct the decoder to generate it (the case of the arbitrary character in the vocabulary);
3. If the output symbol aligns, instruct the decoder to generate _INC_ symbols until the focus-index would reach the corresponding input character, and then generate an _COPY_ symbol;
4. When the sequence is completely generated, instruct the decoder to generate an _EOS_ symbol.

Because English reinflection is fairly simple, we chose an entry from the Romanian dataset for which we present a step-by-step example.

Assume the lemma is “face” (en. “to do”) and is has to be reinflected for the morphological description **V;IND;PST;3;PL;IPFV**. The decoder has to generate word form “făceau” (en. “they were doing”). This means that the inflected form is obtained by replacing the character ‘a’ with the character ‘ă’ and by adding the suffix “au”.

Figure 2 shows the alignments obtained via dynamic programming between the characters of the lemma (up) and the characters of the word form (down). The dashed lines correspond to alignments where the characters in the source and destination are not identical. The final alignments pairs are (according to the straight lines): (0,0), (2,2) and (3,3).

Based on these alignments, the FST symbols generated by the fixed-oracle algorithm are: _COPY_, ‘Ă’, _INC_, _INC_, _COPY_, _INC_, _COPY_, ‘A’, ‘U’, _EOS_. Notice that after copying the first symbol (‘F’) to the output, the oracle immediately generates the vocabulary item ‘Ă’, because it is not aligned with any symbol in the source lemma. However, the next (3rd) symbol in the destination string is aligned with a character in the source string and the index is incremented with two _INC_ commands. The rest of the sequence is generated in a similar fashion.

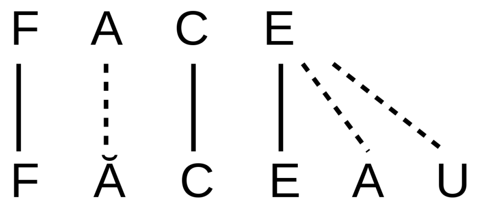

Figure 2: Alignment example

**Note 1:** Fixed-oracle training is known to produce suboptimal results, when compared with dynamic-oracle training. However, we did not have time to experiment with the later mentioned training method and leave this for future work.

## 3 Training details and experimental results

For our implementation is based on DyNET (Neubig et al., 2017), which is a dynamic computation graph network framework. That means that we do not require any padding when we prepare mini-batches.

We evaluated our approach on the data provided during the SIGMORPHON 2018 Shared Task on morphological reinflection (Cotterell et al., 2018). During the evaluation campaign, each language was provided with 3 datasets of different sizes (high, medium and low). Because, neural approaches traditionally require more training data to generalize better, we only built models for the “high” datasets, which were composed of 10K training examples for each language.

Our model was trained using ADAM optimization (Kingma and Ba, 2014), with the default parameters $\alpha = 1e^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We used a mini-batch size of 1K words and we used trained each model until the accuracy on the development set stopped improving for 20 iterations. At the end, we used the best performing model for each languages.

| Language | Acc. | Acc.* | Language | Acc. | Acc.* | Language | Acc. | Acc.* |
|---|---|---|---|---|---|---|---|---|
| adyghe | 92.00 | 97.90 | irish | 81.80 | 86.90 | sanskrit | 76.60 | 94.10 |
| albanian | 95.60 | 97.10 | italian | 90.00 | 91.10 | serbo-croatian | 88.00 | 90.70 |
| arabic | 88.50 | 90.00 | kabardian | 92.00 | 94.00 | slovak | 88.80 | 92.60 |
| armenian | 0.00 | 93.90 | khaling | 44.70 | 45.40 | slovene | 93.40 | 94.10 |
| asturian | 95.40 | 97.00 | kurmanji | 90.40 | 92.10 | sorani | 86.80 | 89.00 |
| azeri | 90.00 | 99.00 | ladin | 86.00 | 90.00 | spanish | 72.70 | 75.00 |
| bashkir | 0.00 | 98.10 | latin | 16.20 | 16.20 | swahili | 98.00 | 99.00 |
| basque | 86.80 | 96.40 | latvian | 92.90 | 97.40 | swedish | 85.20 | 92.10 |
| belarusian | 3.80 | 5.10 | lithuanian | 65.20 | 66.20 | tatar | 95.00 | 97.00 |
| bengali | 99.00 | 99.00 | livonian | 87.00 | 97.00 | turkish | 79.30 | 80.20 |
| breton | 0.00 | 82.00 | lower-sorbian | 93.70 | 96.30 | ukrainian | 90.50 | 94.80 |
| bulgarian | 77.80 | 79.90 | macedonian | 91.90 | 94.60 | urdu | 43.50 | 44.30 |
| catalan | 89.40 | 89.70 | maltese | 4.00 | 91.00 | uzbek | 0.00 | 36.00 |
| classical-syriac | 87.00 | 98.00 | middle-french | 96.70 | 96.30 | venetian | 98.50 | 98.70 |
| crimean-tatar | 94.00 | 96.00 | navajo | 79.00 | 84.00 | votic | 0.00 | 73.00 |
| czech | 90.20 | 92.50 | neapolitan | 0.00 | 40.00 | welsh | 92.00 | 92.00 |
| danish | 91.80 | 94.90 | northern-sami | 90.70 | 93.10 | west-frisian | 0.00 | 93.00 |
| estonian | 93.80 | 97.00 | nor-bokmaal | 88.90 | 92.50 | yiddish | 92.00 | 99.00 |
| faroese | 76.30 | 88.80 | nor-nynorsk | 79.40 | 93.10 | zulu | 73.30 | 74.40 |
| finnish | 87.20 | 92.20 | occitan | 83.00 | 83.00 | dutch | 92.20 | 94.80 |
| friulian | 78.00 | 79.00 | old-armenian | 80.40 | 82.30 | english | 93.80 | 95.10 |
| galician | 89.90 | 91.10 | old-church-slv. | 9.00 | 74.00 | french | 84.30 | 89.80 |
| georgian | 97.80 | 98.40 | old-french | 0.00 | 00.00 | german | 37.40 | 42.50 |
| greek | 81.10 | 85.90 | old-saxon | 54.50 | 54.80 | kannada | 99.00 | 98.00 |
| haida | 96.00 | 93.00 | pashto | 84.00 | 89.00 | north-frisian | 15.00 | 69.00 |
| hebrew | 85.70 | 87.20 | persian | 95.60 | 97.70 | old-english | 28.20 | 30.00 |
| hindi | 89.40 | 90.60 | portuguese | 84.00 | 84.50 | polish | 87.80 | 90.40 |
| hungarian | 79.50 | 86.50 | quechua | 96.80 | 98.30 | russian | 86.80 | 91.40 |
| icelandic | 80.60 | 89.90 | romanian | 82.00 | 88.00 | **Average** | **72.49** | **83.77** |

Table 1: Accuracy figures for all languages in the SIGMORPHON Shared Task 2018

For all languages we used a two-layer encoder with 200 LSTM cells (in each direction - total 400 cells per layer) and a two-layer decoder of 200 unidirectional cells. Each character in the vocabulary is embedded as a 100-dimensional vector. We also use a 100-dimensional embedding size for each unique morphological descriptor.

Table 1 summarizes the testset results for all languages in the SIGMORPHON Challenge 2018. During the official evaluation campaign, our system was affected by a bug which caused all weights belonging to non-recurrent cells to be constant (not trainable during backprop). This issue had a strong negative impact on the results. After this, we retrained our models and we include the unofficial results in the same table, under the "Acc.*" column. For almost all languages, after correcting the bug, the accuracy strongly increased; for Welsh we observed no increase, and only for 2 languages did we observe a less than 1 point decrease (probably due to weight initialization compounded by small models where the LSTMs overcame the fixed random weights of the dense layers). Overall, we observed a strong result increase, from an average of 72.49 to 83.77. For example, for West Frisian where initially the model would not converge (0.00), we now obtain 93.00; similarly, for Armenian, we have gone from 0.00 to 93.9.

## 4 Conclusions

We introduced a specially designed attention-free encoder-decoder model for morphological reinflection. Aside for mitigating standard attention issues, such as repeated or skipped character sequences, this approach allows training simpler models. This is mainly (a) because our model introduces the _COPY_ operation and reduces the representational load of the encoder-decoder model and (b) and because we keep track of the focus-index externally.

Also, we reduce the computational complexity of the model by completely removing calculation involved in the soft attention mechanism ($n * m$ matrix multiplications, where $n$ is the size of the input sequence and $m$ the size of the output sequence).

Moreover, the fact that the decoder does not require taking the previous output and embedding it as input for the next step, demonstrates that there is far less representational overhead involved in generating the output sequence.

As a side note, in our previous experiments with lemmatization, we observed that using this model yields a 2-5% absolute increase in accuracy over the standard soft-attention sequence-to-sequence model.

## Acknowledgments

The research presented here is funded by the Romanian Government through the Executive Agency for Higher Education, Research, Development and Innovation Funding (UEFISCDI), programme "Experimental demonstration project (PED) PED-2016", project ID: PN-III-P2-2.1-PED-2016-1974, contract number 229PED.

## References

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585.

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.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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*, pages 49–57, Vancouver. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara. 2017. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. *arXiv preprint arXiv:1710.08969*.