MED: The LMU System for the SIGMORPHON 2016 Shared Task on Morphological Reinflection

Katharina Kann and Hinrich Schütze
Center for Information & Language Processing
LMU Munich, Germany
kann@cis.lmu.de

Abstract

This paper presents MED, the main system of the LMU team for the SIGMORPHON 2016 Shared Task on Morphological Reinflection as well as an extended analysis of how different design choices contribute to the final performance. We model the task of morphological reinflection using neural encoder-decoder models together with an encoding of the input as a single sequence of the morphological tags of the source and target form as well as the sequence of letters of the source form. The Shared Task consists of three subtasks, three different tracks and covers 10 different languages to encourage the use of language-independent approaches. MED was the system with the overall best performance, demonstrating our method generalizes well for the low-resource setting of the SIGMORPHON 2016 Shared Task.

1 Introduction

In many areas of natural language processing (NLP) it is important that systems are able to correctly analyze and generate different morphological forms, including previously unseen forms. Two examples are machine translation and question answering, where errors in the understanding of morphological forms can seriously harm performance. Accordingly, learning morphological inflection patterns from labeled data is an important challenge.

The task of morphological inflection (MI) consists of generating an inflected form for a given lemma and target tag. Several approaches have been developed for this, including machine learning models and models that exploit the paradigm structure of language (Ahlberg et al., 2015; Dreyer, 2011; Nicolai et al., 2015). A more complex problem is morphological reinflection (MRI). For this, an inflected form has to be found given another inflected form, a target tag and optionally a source tag.

We use the same approach to both MI and MRI: the character-based and language independent sequence-to-sequence attention model called MED, which stands for *Morphological Encoder-Decoder*. To solve the MRI task, we train one single model on all available source-to-target mappings for each language contained in the training set. This enables the encoder-decoder to learn good parameters for relatively small amounts of training data per target tag already, because most MRI patterns occur in many source-target tag pairs. In our model design, what is learned for one pair can be transferred to others.

The most important point for this is the representation we use for MRI. We encode the input as a single sequence of (i) the morphological tags of the source form, (ii) the morphological tags of the target form and (iii) the sequence of letters of the source form. The output is the sequence of letters of the target form. We train a single generic encoder-decoder per language on this representation that can handle all tag pairs, thus making it possible to make efficient use of the available training data.

The SIGMORPHON 2016 Shared Task on Morphological Reinflection covers both, MI and MRI, for 10 languages as well as different settings and MED outperforms all other systems on all subtasks. The given languages, tracks and tasks will be explained briefly now. For further details on the Shared Task please refer to Cotterell et al. (2016).

Languages. In total, the Shared Task covers 10 languages: Arabic, Finnish, Georgian, German, Hungarian, Maltese, Navajo, Russian, Spanish and Turkish. The training and development datasets for Hungarian and Maltese were only released at evaluation time.

Tasks. The Shared Task consists of 3 separate tasks with increasing difficulty: task 1 is supposed to be the easiest and task 3 the hardest. The first task consists of mapping a given lemma and target tag to a target form. Task 2 requires the mapping of a given source form, source tag and target tag to a target form. Finally, task 3 consists of finding a target form for a given source form and source tag only.

Tracks. The Shared Task is split into 3 tracks that differ in the information available. The first track is the standard track and requires the solution for each task to use only the training and development data of the current and all lower-numbered tasks, e.g., to use only the data for tasks 1 and 2 for task 2. The restricted track limits the available training and development data to the data belonging to the current task, i.e., data from lower tasks cannot be used, making it impossible to reduce task 2 to task 1 or task 3 to task 2. Track 3 is the bonus track. In this track, all available data per language can be used, including unlabeled corpora which are provided by the task organizers. However, those vary a lot in length, depending on the language. Therefore, we do not make use of them.

In total, there are 90 combinations of languages, tasks and tracks to solve.

The remainder of this paper is organized as follows: In Section 2, our model for the SIGMORPHON 2016 Shared Task is presented. Next, our method to preprocess and thus extend the training data is explained in detail. In Section 4 the final results on the test data of the Shared Task are presented and discussed. Afterwards, we analyze the contribution of different settings and components to the overall performance of our system in detail. Finally, in Section 6, we give information about prior work on topics related to our system.

This paper is mainly concerned with the implementation and analysis of the system we submitted to the Shared Task. In (Kann and Schütze, 2016), we instead focus on the novel aspects of our new method MED and compare its performance to prior work on other MRI benchmarks.

2 System description

Our system for the Shared Task is an encoder-decoder recurrent neural network (RNN), called MED, which stands for *Morphological Encoder-Decoder*. It will be described in detail in this Section.

2.1 Neural network model

Our model is based on the network architecture proposed by Bahdanau et al. (2014) for machine translation.[1] The authors describe the model in detail; unless we explicitly say so in the description of our model below, we use the same network configuration as they do.

Bahdanau et al. (2014)'s model is an extension of the recurrent neural network (RNN) encoder-decoder developed by Cho et al. (2014) and Sutskever et al. (2014). The encoder of the latter consists of a gated RNN (GRU) that reads an input sequence of vectors x and encodes it into a fixed-length context vector c, computing hidden states h_t and c by

$$h_t = f(x_t, h_{t-1}) \tag{1}$$

and

$$c = q(h_1, ..., h_{T_x}) \tag{2}$$

with nonlinear functions f and q. The decoder uses the context vector to predict the output y:

$$p(y) = \prod_{t=1}^{T_y} p(y_t | \{y_1, ..., y_{t-1}\}, c) \tag{3}$$

with $y = (y_1, ..., y_{T_y})$ and each conditional probability being modeled with an RNN as

$$p(y_t | \{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c) \tag{4}$$

where g is a nonlinear function and s_t is the hidden state of the RNN.

Bahdanau et al. (2014) proposed an attention-based version of this model that allows different vectors c_t for each step by automatic learning of an alignment model. They further made the encoder bidirectional. In their model each hidden state h_j at time step j does not only depend on the preceding, but also on the following input:

$$h_j = \left[\overrightarrow{h_j^T}; \overleftarrow{h_j^T}\right]^T \tag{5}$$

[1]Our implementation of MED is based on `https://github.com/mila-udem/blocks-examples/tree/master/machine_translation`.

The formula for $p(y)$ changes accordingly:

$$p(y) = \prod_{t=1}^{T_y} p(y_t|\{y_1, ..., y_{t-1}\}, x) \tag{6}$$

$$= \prod_{t=1}^{T_y} g(y_{t-1}, s_t, c_t) \tag{7}$$

with s_t being an RNN hidden state for time t and c_t being the weighted sum of the annotations $(h_1, ..., h_{T_x})$ produced by the encoder:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{8}$$

The attention weights α_{ij} are calculated for each h_j as

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{9}$$

with

$$e_{ij} = a(s_{i-1}, h_j) \tag{10}$$

a is parametrized as a feedforward neural network and trained jointly with all other components.

More theoretical background is given in (Bahdanau et al., 2014) and a system overview can be seen in Figure 1.

The final model is a multilayer network with a single maxout (Goodfellow et al., 2013) hidden layer that computes the conditional probability of each element in the output sequence (a character in our case, (Pascanu et al., 2014)). As MRI is less complex than machine translation, we reduce the number of hidden units and the embedding size. After initial experiments, we fixed the hyperparameters of our system and did not further adapt them to a specific task or language. Encoder and decoder RNNs have 100 hidden units each. For training, we use stochastic gradient descent, Adadelta (Zeiler, 2012) and a minibatch size of 20. We initialize all weights in the encoder, decoder and the embeddings except for the GRU weights in the decoder with the identity matrix as well as all biases with zero (Le et al., 2015). We train all models for 20 iterations for all combinations of track and task where we cannot extend the training data with our method described in the next section. Otherwise, we train for 10 epochs.[2] We settled on this number in early experimentation because training usually converged before that limit.

MED is an ensemble of five RNN encoder-decoders. The final decision is made by majority voting. In case of a tie, the answer is chosen randomly among the most frequent predictions.

2.2 Input and output format

We define the alphabet Σ_{lang} as the set of characters used in the application language. As each tag combination which describes a source or target form consists of one or more subtags, e.g., "number" or "case", we further define Σ_{src} and Σ_{trg} as the set of morphological subtags seen during training as part of the source tag or the target tag, respectively. Finally, we define S_{start} and S_{end} to be a start and an end symbol. Then each input of our system is of the format $S_{start}\Sigma^{+}_{src}\Sigma^{+}_{trg}\Sigma^{+}_{lang}S_{end}$. In the same way, we define the output format as $S_{start}\Sigma^{+}_{lang}S_{end}$.

For example, a valid input for German would be *<w> IN=pos=ADJ IN=case=GEN IN=num=PL OUT=pos=ADJ OUT=case=ACC OUT=num=PL i s o l i e r t e r </w>*. The corresponding system output should be *<w> i s o l i e r t e </w>*.[3]

3 Data and training

3.1 Training data enhancement

Since the Shared Task models a low-resource setting, a way to enhance the given training data is highly desirable. We apply three different methods for this, depending on the track and, therefore, depending on the information available. Even though the training data enhancer could be used to increase the amount of available data for other models as well, we expect it to be especially effective with MED. This is due to the fact that MED is able to reuse information from any combination of input and output tag for any other tag pair.

Restricted track. In the restricted track, only training and development data of the respective task and language can be used. This means that there is less information available than in the other two tracks. Therefore, in this track we can only use a very basic enhancement method and we can

[2] For extended data in Maltese we trained only for 6 epochs, due to time constraints.

[3] For task 1 in the restricted and standard track and task 3 throughout all tracks, no source tag is given and we only have one tag combination in the input. Therefore, we do not prepend *IN=* or *OUT=* to the tags. However, internally, this does not make a difference for the model.

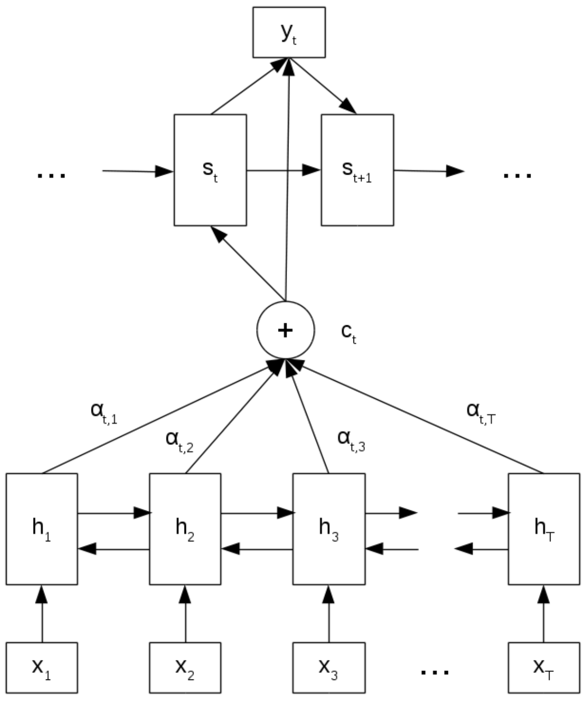

Figure 1: System overview. The input x consists of characters as well as input and output tags. The output y consists of characters only.

only apply it to task 2. The idea the method is based on is that task 2 is symmetric. As described before, the task consists of mapping a triplet of source tag, source form and target tag to a target form. To double the training data it is sufficient to switch the information and thus create a new sample, mapping from target tag, target form and source tag to the source form.

Standard track. The training data enhancement for the standard track combines information from task 1 and task 2 and can therefore, following the Shared Task rules, be used for task 2 and task 3, as only data from lower tasks needs to be accessed. The idea of our enhancement method is that each word form belongs to a certain paradigm which in turn belongs to one single lemma. Therefore, when knowing the lemmas of words, we can group them into paradigms. When having more than one word per paradigm, we can infer the information that all of them can be inflected into each other and thus use them to create new samples. Knowing this, we use task 1 training data to make groups of lemmas and word forms belonging to the same paradigm, keeping the tags. Then, we add all information from task 2 and, knowing that source form and target form always belong to the same lemma, we add both forms with their tags to a group whenever one of them is already in there.[4] Afterwards, we build all combinations of word pairs of each paradigm and, by doing so, create new training data.

This method could be applied even if there was absolutely no overlap between the lemmas in task 1 and task 2. However, it would then be necessary to train a lemmatizer on task 1 data first and lemmatize all words to sort them into paradigms. When doing this, accuracy could be improved by only accepting predictions with a strong confidence and by only accepting new words for a paradigm if the source and the target form of a sample have the same lemma prediction.

Bonus track. In the bonus track, our training data enhancement can also be used for task 1. In order to do so, we first apply the same method as for the standard track to produce the extended training data. However, we additionally change the input format for task 1 such that it resembles the task 2 input, using *LEMMA* as the input tag. In this way, we can apply the task 2 model to task 1 such that task 1 is able to benefit from the additional data as well.

3.2 Description of the final training data

Depending on the complexity of the language and the structure of the datasets we end up with a different amount of final training samples for each language, even though we start with nearly identical training set sizes. We show the final number of samples for task 2 in different tracks in Table 1. As can be seen, the training data enhancement increases the number of samples by a factor between 10 and 80. Out of all languages, the enhancer has the smallest effect for Finnish. Most additional samples are created for Maltese.

3.3 Training

For each of the 10 languages we train one ensemble for each task of the restricted track as well as for each of tasks 2 and 3 of the bonus track. We do not train a separate model for task 1, due to the fact that the same model can be applied to both task 1 and task 2 of the bonus track. In total, we train 50 ensembles, consisting of 250 single models. For our setting, task 1 of the standard track is the same as for the restricted track, while tasks 2 and 3 are the same as for the bonus track.

[4]As for none of the languages task 3 contained new word forms, we did not consider task 3 data here.

Dataset	T2, given no. samples	T2, restricted no. samples	factor	T2, standard no. samples	factor
Arabic	14,400	28,800	2	458,814	32
Finnish	14,400	28,800	2	116,206	8
Georgian	14,400	28,800	2	196,396	14
German	14,400	28,800	2	166,148	12
Hungarian	21,600	43,200	2	643,630	30
Maltese	21,600	43,200	2	1,629,446	75
Navajo	14,385	28,770	2	160,332	11
Russian	14,400	28,800	2	129,302	9
Spanish	14,400	28,800	2	211,030	15
Turkish	14,400	28,800	2	392,136	27

Table 1: Number of training samples for task 2 without (given) and with the training data enhancer (restricted and standard track) together with the factor by which the size of the training set increased. Note that the samples for task 2 in the standard track are the same as the samples for task 1 in the bonus track.

Language	Task 1	Task 2	Task 3
Arabic	95.47%	97.38%	96.52%
Finnish	96.80%	97.40%	96.56%
Georgian	98.50%	99.14%	98.87%
German	95.80%	97.45%	95.60%
Hungarian	99.30%	99.67%	99.50%
Maltese	88.99%	88.17%	87.83%
Navajo	91.48%	96.64%	96.20%
Russian	91.46%	91.00%	89.91%
Spanish	98.84%	98.74%	97.96%
Turkish	98.93%	97.94%	99.31%

Table 2: Exact-match accuracy per language for the standard track of the SIGMORPHON 2016 Shared Task.

Language	Task 1	Task 2	Task 3
Arabic	98.25%	97.38%	96.25%
Finnish	97.30%	97.40%	96.56%
Georgian	99.20%	99.14%	98.87%
German	97.38%	97.45%	95.60%
Hungarian	99.69%	99.67%	99.50%
Maltese	88.53%	88.17%	87.83%
Navajo	98.03%	96.64%	96.20%
Russian	92.15%	91.00%	89.91%
Spanish	99.05%	98.74%	97.96%
Turkish	97.49%	97.94%	99.31%

Table 4: Exact-match accuracy per language for the bonus track of the SIGMORPHON 2016 Shared Task.

Language	Task 1	Task 2	Task 3
Arabic	95.47%	91.09%	82.80%
Finnish	96.80%	96.81%	93.18%
Georgian	98.50%	98.50%	96.21%
German	95.80%	96.22%	92.41%
Hungarian	99.30%	99.42%	98.37%
Maltese	88.99%	86.88%	84.25%
Navajo	91.48%	97.81%	83.50%
Russian	91.46%	90.11%	87.13%
Spanish	98.84%	98.45%	96.69%
Turkish	98.93%	98.38%	95.00%

Table 3: Exact-match accuracy per language for the restricted track of the SIGMORPHON 2016 Shared Task.

For each task of the restricted track we train a separate model for 20 epochs. For the bonus track we reduce the number of epochs to 10, because we have much more training data. For Maltese, we reduce it even further to 6 epochs.

Because we do not tune any hyperparameters, we combine the original training and development sets to one big training set. The numbers reported in Table 1 are considering this big dataset.

4 Results on the Shared Task test data

Tables 2, 3 and 4 list the official final results of MED for the SIGMORPHON 2016 Shared Task. Table 2 shows the results of the standard track for which systems are allowed to access the data of the respective task and all lower numbered tasks. Therefore, we can apply our training data extension to tasks 2 and 3, but not to task 1. Because of this, the two higher tasks have the same scores as in the bonus track: we effectively give the same answers. Task 1, in turn, is the same for the standard and the restricted track, leading to the same numbers in Tables 2 and 3.

For ease of exposition, we will mostly compare the restricted and the bonus track as the standard track can be considered a mixture of those two. For most tasks and languages the accuracy is higher in the bonus than in the restricted track. This is easy to explain as MED has more data to train on (task 1 information for tasks 2 and 3 and task 2 information for task 1). The exception is Navajo: For task 2 the accuracy is higher in the bonus track than in the restricted track. We leave an investigation of this for future work.

Our training data enhancer – which is the only difference between the bonus and the restricted track as we do not use the provided unlabeled corpora – is clearly effective: For Arabic, for example, it leads to 13.72% improvement in performance for task 3. For Turkish, the accuracy for task 3 increases by 4.31%. Those are also the languages for which the training data enhancement was very effective as can be seen in Table 1. That Maltese does not improve so much even though we use a lot more training data is most likely due to the shorter training: we trained only for 6 epochs instead of 10, because of time constraints.

As expected, the scores for task 3 are worse than or at most comparable to the scores for task 2 in all tracks. This is due to the fact that task 3 does not provide a source tag, so less information is available. However, it seems that this information was not much needed as the improvement when adding it is minor. The better result for task 3 for Turkish compared to task 2 in the bonus track may be due to randomness during training – like the order of samples in the training data – as it is below 1.5%.

It may be surprising at first that the results for task 1 are not always better than the results for task 2. This is the case, for example, in the restricted track for Finnish, Georgian, Hungarian and Navajo. As the organizers describe on the Shared Task's homepage, they expect task 1 to be the easiest. Our guess would be that the model has more information in total for task 2 as more forms are given per paradigm. Additionally, task 2 is symmetric; this makes it possible to use twice the training data, as described in Section 3.

5 System Analysis

To analyze which design choices are important and how they influence the performance of MED we conduct several experiments, always keeping all but the investigated design choice fixed to the settings described in Section 2. To make the experiments clearer, we limit them to one combination of task, track and language: Unless mentioned otherwise, we perform all experiments described in this section on task 2 of the restricted track for Russian. For the experiments in this section, the system is trained on training data only and evaluated on the development set. The training data enhancement is not used in this analysis.

5.1 Analysis 1: Number of hidden units in encoder and decoder

In its original configuration MED has 100 hidden units in both the encoder and the decoder. This number was found to be good during initial experiments. However, we want to investigate how the number of hidden units in the RNNs can effect the final accuracy on an MRI task. Therefore, we train one ensemble for each of 50, 100, 200 and 400 hidden units. To reduce the number of possible different options and because it agrees with MED's original configuration, we define the numbers of hidden units in encoder and decoder to be equal.

Number of hidden units	Exact-match accuracy
50	86.2%
100	88.4%
200	87.2%
400	87.3%

Table 5: Performance of MED for different numbers of hidden units in the encoder and decoder.

Embedding size	Exact-match accuracy
100	86.7%
200	87.3%
300	88.4%
400	90.0%
500	90.3%

Table 6: Performance of MED for different embedding dimensions in the encoder and decoder.

The evaluation in Table 5 shows that the best accuracy is obtained for 100 hidden units. Lower results for fewer hidden units indicate that the model does not have enough capacity to learn the patterns in the data well. Lower results for more hidden units indicate that the model is overfitting the training data.

5.2 Analysis 2: Size of the embeddings

We chose 300 to be the size of the character and tag embeddings in our model for the Shared Task. In this analysis, we want to systematically investigate how MED performs for different embedding sizes for the encoder and decoder embeddings. We train the model with embeddings of the sizes 100, 200, 300, 400 and 500 and report the resulting accuracies in Table 6.

The results show that the bigger the embeddings get the more the perfomance improves. The best accuracy is reached for 500-dimensional embeddings, i.e., the biggest embeddings in this analysis. This suggests that we might have improved our final results in the Shared Task even further by using embeddings of a higher dimensionality. However, this is also a trade-off between a gain in accuracy and longer training time. Keeping in mind that we had to train many single models, 300 was a reasonable choice for the embedding size, with only 1.9% loss of perfomance compared to 500-dimensional embeddings.

Initialization	Exact-match accuracy
Identity	90.5%
Identity + orthogonal	88.4%
Gaussian + orthogonal	89.7%

Table 7: Performance of MED for different initialization types.

5.3 Analysis 3: Initialization

For the Shared Task, most weights of MED are initialized with the identitiy matrix. An exception to this are the weights in the decoder GRU which are initialized using a random orthogonal matrix. All biases are initialized to zero. We now compare how MED's final performance depends on the type of initialization. For this, we train two additional models: (i) we initialize all weights with the identitiy matrix and (ii) we initialize all weights except for the weights in the decoder GRU from a Gaussian distribution. The weights in the decoder GRU are again initialized with a random orthogonal matrix.

The final accuracy of the three models can be seen in Table 7. The random intialization leads to better results than intitializing with the identity matrix together with a random orthogonal matrix. However, the highest accuracy is reached by initializing *all* weights with identity matrices. In fact, the results are 2.1% better than MED's original performance. Thus, we would recommend this initialization for future use of our model.

5.4 Analysis 4: One embedding per tag vs. one embedding per tag combination

To keep the model flexible to handle tag combinations not present in the training set, we split each tag combination into single tags, e.g., *pos=ADJ,case=ACC,gen=FEM,num=SG* becomes *pos=ADJ*, *case=ACC*, *gen=FEM* and *num=SG* with each part having its own embedding which is fed into the model.

We now compare to the performance of a representation in which tags are "fused" and each tag combination has only one single embedding. As this is one of the most important design choices for MED, we do this analysis for several languages and additionally report the number of tag combinations that are not seen during training.

Table 8 shows that unknown tag combinations are generally not a problem with the exception of Maltese. Nevertheless, there is a considerable decrease in performance. The difference is especially big for languages with a lower performance like Arabic, Maltese, Navajo and Russian. Languages with a general high accuracy do not lose much accuracy when using one embedding per tag combination. We hypothesize that the patterns of these languages are easy enough to even be learned with a harder representation. Overall, it seems that our representation with split-up tag combinations is the better choice for MRI and might even be a key component for MED's success in the Shared Task.

Language	MED	MED-tag-comb.	Unk.
Arabic	88.8%	83.4%	0
Finnish	95.6%	95.2%	1
Georgian	97.3%	95.6%	0
German	95.1%	93.5%	1
Hungarian	99.3%	99.3%	0
Maltese	85.7%	77.1%	151
Navajo	91.1%	83.4%	1
Russian	88.4%	86.8%	1
Spanish	97.5%	97.0%	0
Turkish	97.6%	95.9%	2

Table 8: Exact match accuracy for the standard representation (MED) as well as the representation with one embedding per tag combination (MED-tag-comb) per language. The last column shows the number of samples that contain tag combinations that appear in dev but not in train, either for the source or the target form.

Tag order type	Exact-match accuracy
MED	88.4%
MED-perm	86.4%

Table 9: Performance of MED when training on samples with tags in always the same order (MED) and samples where the tags are permuted inside each combination (MED-perm).

5.5 Analysis 5: The order of tags

In the representation we feed to MED, the order of single tags inside a tag combination is always fixed. We now investigate how much influence this has on the final performance of the model; i.e., we ask: is MRI harder or easier to learn if we permutate the morphological tags? For this analysis, we randomly shuffle the tags of each combination in the training and development data (while still using the development set for testing).

Table 9 shows that learning seems to be easier for non-permuted tags. Indeed, when keeping the order of tags fixed, the system performance is 2% better than for the random tag order. However, the difference is not big. This might actually be different for languages other than Russian as we did not investigate from a linguistic point of view if the order matters contentwise for any of the languages.

6 Related Work

Prior work on morphology includes morphological segmentation (Harris, 1955; Hafer and Weiss, 1974; Déjean, 1998), different approaches for MRI (Ahlberg et al., 2014; Durrett and DeNero, 2013; Eskander et al., 2013; Nicolai et al., 2015). and work on morphological tagging and lemmatization (Müller et al., 2015).

RNN encoder-decoder models, gated RNNs in general as well as LSTMs were applied to several NLP tasks including some on morphology like morphological segmentation (Wang et al., 2016) during the last years. Other tasks they proved to be useful for are machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014), parsing (Vinyals et al., 2015) or speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013).

The most similar work to ours was probably the one by Faruqui et al. (2015). Indeed, MED's design is very close to their model. However, they trained one network for every tag pair; this can negatively impact performance in a setting with limited training data like the SIGMORPHON 2016 Shared Task. In contrast, we train a single model for each language. This radically reduces the amount of training data needed for the encoder-decoder because most MRI patterns occur in many tag pairs, so what is learned for one can be transferred to others. In order to model all tag pairs of the language together, we introduce an explicit morphological representation that enables the attention mechanism of the encoder-decoder to generalize MRI patterns across tag pairs.

7 Conclusion

In this paper we described MED, our system for the SIGMORPHON 2016 Shared Task on Morphological Reinflection as well as a training data enhancement method based on paradigms. MED is a powerful character-based encoder-decoder RNN and its architecture is completely language-independent, such that we trained the models for all 10 languages of the Shared Task using the same hyperparameters. MED establishes the state of the art for the SIGMORPHON 2016 Shared Task, scoring first in all of the 90 subtasks of the final evaluation.

Furthermore, we presented an extended analysis, evaluating different design choices for MED. The results show that most of our initial settings were good choices, especially the representation of morphological tags. However, it might be possible to further improve MED's performance increasing the size of the used embeddings and choosing another initialization.

Acknowledgments

We are grateful to MILA (`https://mila.umontreal.ca`) for making their neural machine translation model available to us. We further acknowledge the financial support of Siemens for this research.

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proc. of EACL*.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proc. of NAACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proc. of the 2016 Meeting of SIGMORPHON*.

Hervé Déjean. 1998. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proc. of the Joint Conferences on New Methods in Language Processing and CoNLL*.

Markus Dreyer. 2011. *A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proc. of HLT-NAACL*.

Ramy Eskander, Nizar Habash, and Owen Rambow. 2013. Automatic extraction of morphological lexicons from morphologically annotated corpora. In *Proc. of EMNLP*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2015. Morphological inflection generation using character sequence to sequence learning. *arXiv preprint arXiv:1512.06110*.

Ian Goodfellow, David Warde-farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proc. of ICML*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc. of ICASSP*.

Margaret A Hafer and Stephen F Weiss. 1974. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.

Zellig S Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proc. of ACL*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proc. of EMNLP*.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proc. of NAACL*.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. In *Proc. of ICLR*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *Proc. of AAAI*.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.