# Improving Sequence to Sequence Learning for Morphological Inflection Generation: The BIU-MIT Systems for the SIGMORPHON 2016 Shared Task for Morphological Reinflection

**Roee Aharoni** and **Yoav Goldberg**
Computer Science Department
Bar Ilan University
roee.aharoni,yoavgo@gmail.com

**Yonatan Belinkov**
CSAIL
MIT
belinkov@mit.edu

## Abstract

Morphological reinflection is the task of generating a target form given a source form and the morpho-syntactic attributes of the target (and, optionally, of the source). This work presents the submission of Bar Ilan University and the Massachusetts Institute of Technology for the morphological reinflection shared task held at SIGMORPHON 2016. The submission includes two recurrent neural network architectures for learning morphological reinflection from incomplete inflection tables while using several novel ideas for this task: morpho-syntactic attribute embeddings, modeling the concept of templatic morphology, bidirectional input character representations and neural discriminative string transduction. The reported results for the proposed models over the ten languages in the shared task bring this submission to the second/third place (depending on the language) on all three sub-tasks out of eight participating teams, while training only on the Restricted category data.

## 1 Introduction

Morphological inflection, or reinflection, involves generating a target (surface form) word from a source word (e.g. a lemma), given the morpho-syntactic attributes of the target word. Previous approaches to automatic inflection generation usually make use of manually constructed Finite State Transducers (Koskenniemi, 1983; Kaplan and Kay, 1994), which are theoretically appealing but require expert knowledge, or machine learning methods for string transduction (Yarowsky and Wicentowski, 2000; Dreyer and Eisner, 2011; Durrett and DeNero, 2013; Hulden et al., 2014; Ahlberg et al., 2015; Nicolai et al., 2015). While these studies achieved high accuracies, they also make specific assumptions about the set of possible morphological processes that create the inflection, and require feature engineering over the input.

More recently, Faruqui et al. (2016) used encoder-decoder neural networks for inflection generation inspired by similar approaches for sequence-to-sequence learning for machine translation (Bahdanau et al., 2014; Sutskever et al., 2014). The general idea is to use an encoder-decoder network over characters, that encodes the input lemma into a vector and decodes it one character at a time into the inflected surface word. They factor the data into sets of inflections with identical morpho-syntactic attributes (we refer to each such set as a factor) and try two training approaches: in one they train an individual encoder-decoder RNN per factor, and in the other they train a single encoder RNN over all the lemmas in the dataset and a specific decoder RNN per factor.

An important aspect of previous work on learning inflection generation is the reliance on complete inflection tables – the training data contains all the possible inflections per lemma. In contrast, in the shared task setup (Cotterell et al., 2016) the training is over partial inflection tables that mostly contain only several inflections per lemma, for three different sub-tasks: The first requires morphological inflection generation given a lemma and a set of morpho-syntactic attributes, the second requires morphological re-inflection of an inflected word given the word, its morpho-syntactic attributes and the target inflection's attributes, and the third requires re-inflection of an inflected word given only the target inflection attributes. The datasets for the different tasks are available on the

*Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 41–48,
Berlin, Germany, August 11, 2016. ©2016 Association for Computational Linguistics

shared task's website.[1]

The fact that the data is incomplete makes it problematic to use factored models like the ones introduced in (Faruqui et al., 2016), as there may be insufficient data for training a high-quality model per factor of inflections with identical morpho-syntactic attributes. For example, in the shared task dataset the training data usually contains less than 100 training examples on average per such factor. Moreover, when the data is factored this way, no information is shared between the different factors even though they may have identical inflection rules.

We propose two neural network architectures for the task. The first, detailed in Section 2, departs from the architecture of (Faruqui et al., 2016) by extending it in three novel ways: representing morpho-syntactic attributes, template-inspired modeling, and bidirectional input character representations. The second, described in Section 3, is based on an explicit control mechanism we introduce while also making use of the three extensions mentioned above. Our experimental evaluation over all 10 languages represented in the shared task brings our models to the second or third place, depending on the language.

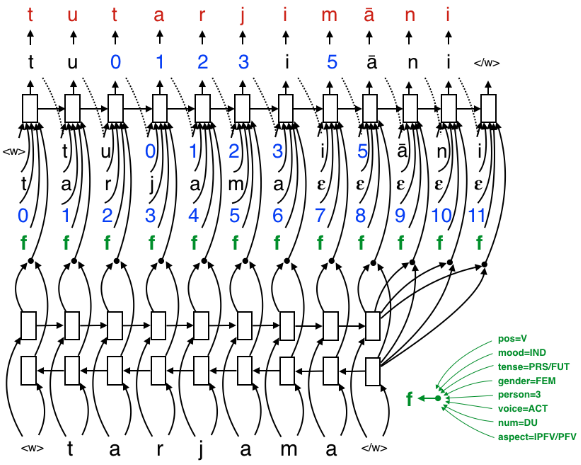
## 2 First Approach: Morphological Sequence to Sequence Architecture

Our first proposed architecture is a Morphological Sequence to Sequence (MS2S) architecture, illustrated in Figure 1. It incorporates several novel components in the sequence-to-sequence learning paradigm, as discussed below.

### 2.1 Morpho-Syntactic Attribute Embeddings

We seek to train models over larger amounts of examples, rather than on factors that strictly contain examples that share all the morpho-syntactic attributes. To do so, instead of factoring the data by the attributes we feed the attributes into the network by creating a dense embedding vector for every possible attribute/value pair (for example, gender=FEM and gender=MASC will each have its own embedding vector). The attribute embeddings for each input are then concatenated and added as parameters to the network while being updated during training similarly to the character embeddings. This way, information can be shared across inflections with different morpho-syntactic attributes, as they are trained jointly, while the attribute embeddings help discriminate between different inflection types when needed. This can be seen in Figure 1, where $f$ is the vector containing a concatenation of the morpho-syntactic attribute embeddings.

Figure 1: The Morphological Sequence to Sequence (MS2S) network architecture for predicting an inflection template given the Arabic lemma *tarjama* and a set of morpho-syntactic attributes. A round tip expresses concatenation of the inputs it receives.



While this approach should allow us to train a single neural network over the entire dataset to predict all the different inflection types, in practice we were not able to successfully train such a network. Instead, we found a middle ground in training a network per part-of-speech (POS) type. This resulted in much fewer models than in the factored model, each using much more data, which is essential when training machine learning models and specifically neural networks. For example, on the Arabic dataset of the first sub task (inflection generation from lemma to word) this reduced the amount of trained models from 223 with an average of 91 training examples per model, to only 3 models (one per POS type - verb, noun, adjective) with an average of 3907 training examples per model.

### 2.2 Morphological Templates

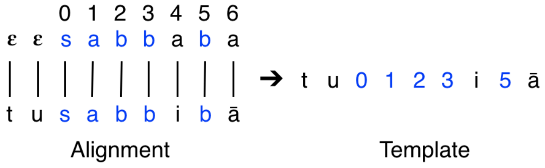
We bring the idea of morphological templates into the model: instead of training the network to predict only a specific inflection character at each step given a lemma and a set of morpho-syntactic features, we train the network to either predict a char-

[1] http://ryancotterell.github.io/sigmorphon2016/

acter from the vocabulary or *to copy* a character at a given position in the input sequence. This enables the network to produce a sequence that resembles a morphological template which can be instantiated with characters from the input to produce the correct inflection. While at train time we encourage the network to perform copy operations when possible, at prediction time the network can decide whether to copy a character from the input by predicting its location in the input or to generate a preferred character from the vocabulary. For example, for the Arabic lemma *tarjama* and a set of morpho-syntactic attributes the network will output the sequence ”tu0123i5āni” which can be instantiated with the lemma into the correct inflection, *tutarjimāni*, as depicted in Figure 1.

Intuitively, this method enables the learning process to generalize better as many different examples may share similar templates – which is important when working with relatively small datasets. We saw indeed that adding this component to our implementation of a factored model similar to (Faruqui et al., 2016) gave a significant improvement in accuracy over the Arabic dataset: from 24.04 to 78.35, while the average number of examples per factor was 91.

To implement this, for every given pair of input and output sequences in the training set we need to produce a parameterized sequence which, when instantiated with the input sequence, creates the output sequence. This is achieved by running a character level alignment process on the training data, which enables to easily infer the desired sequence from every input-output sequence alignment. For example, given the input sequences *sabbaba* and output sequence *tusabbibā* with the induced alignment $\epsilon\epsilon$sabbaba-tusabbibā, we produce the expected output: tu0123i5ā, as depicted in the next figure:



We performed the alignment process using a Chinese Restaurant Process character level aligner (Sudoh et al., 2013) as implemented in the shared task baseline system.[2]

## 2.3 Bidirectional Input Character Representation

Instead of feeding the decoder RNN at each step with a fixed vector that holds the encoded vector for the entire input sequence like Faruqui et. al. (2016), we feed the decoder RNN at each step with a Bi-Directional Long-Short Term Memory (BiLSTM) representation (Graves and Schmidhuber, 2005) per character in the input along with the character embedding learned by the network. The BiLSTM character representation is a concatenation of the outputs of two LSTMs that run over the character sequence up to the current character, from both sides. This adds more focused context when the network predicts the next inflection output, while still including information form the entire sequence due to the bidirectional representation.

## 2.4 MS2S Decoder Input

For every step $i$ of the decoder RNN for this setup, the input vector is a concatenation of the following:

1. $BiLSTM_i$ – The bidirectional character embedding for the $i$th input character (if $i$ is larger than the length of the input sequence, the embedding of the last input character is used).

2. $c_i$ – The character embedding for the $i$th input character. If $i$ is larger than the length of the input sequence, an embedding of a special $\mathcal{E}$ symbol is used, similarly to (Faruqui et al., 2016).

3. $i$ – A character embedding for the current step index in the decoder. In the first step this will be an embedding matching to ’0’, in the second step it will an embedding matching to ’1’ etc. These are the same index embeddings used to model copy actions from a specific index.

4. $o_{i-1}$ – The feedback input, containing the embedding of the prediction (either a character or an integer representing an index in the input) from the previous decoder RNN step.

5. $f$ – The vector containing the concatenation of the morpho-syntactic attribute embeddings.

[2] https://github.com/ryancotterell/sigmorphon2016/tree/master/src/baseline

# 3 Second Approach: The Neural Discriminative String Transducer Architecture

The second approach is based on a Neural Discriminative String Transducer (NDST), a novel neural network architecture that models which specific part of the input sequence is relevant for predicting the next output character at a given time. This is done by maintaining a state consisting of an input sequence position (the input pointer) and an output sequence position (the output pointer), which are controlled by the decoder. This approach can be seen as a more focused replacement to the general attention mechanism of Bahdanau et. al. (2014), tailored to the usually monotonic behavior of the output sequence with respect to the input sequence in the morphological reinflection task. An example for using this architecture is available in Figure 2.

## 3.1 NDST Decoder Input

For every step $i$ in the NDST decoder RNN, the input vector is a concatenation of the following:

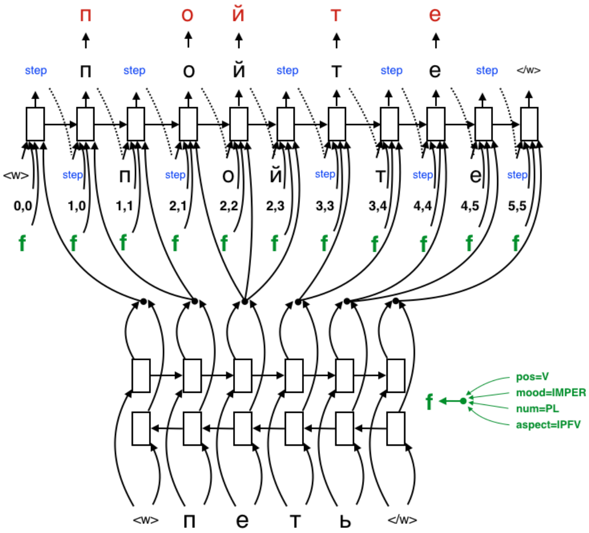
1. $p_{input}$ – The input pointer, holding the embedding that represents the position of the current pointed input sequence element. When $i = 0$, this is initialized with the embedding that stands for the position of the first element in the input. Every time the network outputs the "step" symbol, $p_{input}$ is promoted by setting it with the embedding that represents the next input sequence position.

2. $p_{output}$ – The output pointer, a character embedding representing the next position in the output sequence to be generated. When $i = 0$, this is initialized with the embedding that stands for the position of the first element in the input. Every time the network outputs a symbol other than the "step" symbol, $p_{output}$ is promoted by setting it with the embedding for the next output sequence position.

3. $BiLSTM_{p_{input}}$ – The bidirectional character embedding for the input character currently pointed by $p_{input}$.

4. $o_{i-1}$ – The feedback input, containing the embedding of the prediction (either a character, an integer representing an index in the input, or the "step" symbol) from the previous decoder RNN step.

Figure 2: The Neural Discriminative String Transducer (NDST) architecture for predicting an inflection template given a lemma and a set of morpho-syntactic attributes.



5. $f$ – The vector containing the concatenation of the morpho-syntactic attribute embeddings.

To train an NDST network, for every input and output sequence in the training data we should have a sequence of actions (of three types – either a specific character prediction, an index to copy from or a "step" instruction) that when performed on the input sequence, produces the correct output sequence. To get the correct instruction sequences in train time we first run a character level alignment process on the training data, similarly to the MS2S model. Once we have the character level alignment per input-output sequence pair, we deterministically infer the sequence of actions that results in the desired output by going through every pair of aligned input-output characters in the alignment. If the input and output characters in the aligned pair are not identical, we produce the new output character. If the input and output characters in the aligned pair are identical we produce a copy action from the input character location. After that, if the next output character is not the epsilon symbol as seen in the alignment in Figure 2.2 we also produce a "step" action. We train the network to produce this sequence of actions when given the input sequence and the set of morpho-syntactic attributes matching the desired inflection.

## 4 Experimental Details

### 4.1 Submissions

The shared task allowed submissions in three different tracks: Standard, which enabled using data from lower numbered tasks in addition to the current task data; Restricted, which enabled using only the current task's data; and Bonus, which enabled using the Standard track datasets and an additional monolingual corpus supplied by the organizers.

We submitted two systems to the shared task, both in the Restricted track: The first, named BIU/MIT-1, used the MS2S architecture as described previously and participated in all three sub-tasks. Notice that for the 3rd task, the input is identical to the first task so it does not require changes in the network architecture. To use the MS2S network for the second task we concatenated the source and target morpho-syntactic attribute embeddings and used that vector as the $f$ vector mentioned previously. The output from this system was 5-best lists, meaning 5 predictions for each input. To produce the 5-best list we perform beam search over the MS2S model, which is trained greedily without such search procedure.

The second system, named BIU/MIT-2, used the NDST architecture and participated only in the first and second sub-tasks. This system did not use beam search, producing only one guess per input. Again, to use the NDST architecture for the second task we simply concatenated the input and output morpho-syntactic attribute embeddings.

### 4.2 Training, Implementation and Hyper Parameters

To train our systems, we used the train portion of the dataset as-is and submitted the model which performed best on the development portion of the dataset, without conducting any specific preprocessing steps on the data. We trained our networks for a maximum of 300 epochs over the entire training set or until no improvement on the development set has been observed for more than 100 epochs. The systems were implemented using pyCNN, the python wrapper for the CNN toolkit.[3] In both architectures we trained the network by optimizing the expected output sequence likelihood using cross-entropy loss. For optimization we used ADAM (Kingma and Ba, 2014) with no regularization, and the parameters set as $\alpha = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. In all architectures we used the CNN toolkit implementation of an LSTM network with two layers, each having 200 entries. The character embeddings were also vectors with 200 entries, and the morpho-syntactic attribute embeddings were vectors of 20 entries. When using beam search we used a beam width of 5.

## 5 Results

While developing our systems we measured our performance on the development set with respect to two baselines: the shared task baseline system (ST-Base) inspired by (Nicolai et al., 2015; Durrett and DeNero, 2013), and the factored sequence to sequence baseline (Fact.) similar to the one introduced in (Faruqui et al., 2016). On the test set, our systems ranked second or third out of eight groups in the shared task (depending on the language). The best participating system, LMU-1/2 (Kann and Schütze, 2016) relied on a single encoder-decoder model with attention (Bahdanau et al., 2014) per language, with several improvements like performing prediction using majority voting over an ensemble of five models. In contrast, our first system did not use an explicit attention mechanism and is composed of 3 models per language (one per POS type) without using ensembling. We compare our system to the best system on the test set.

The results for the first task are shown in Table 1, measuring aggregated accuracy across all POS tags. On the development set, our models surpassed both baselines significantly and were competitive with each other, as the MS2S model gained the best aggregated accuracy results on all languages but Russian and Finnish, where the NDST model was better. On the test set, similar results are shown: the MS2S model gives higher accuracies except for Russian, Navajo and Maltese where the NDST model was superior.

For the second task, we measured performance only with respect to ST-Base as can be seen in Table 2. On the development set, the NDST model outperformed the baseline and the MS2S model for all languages but Georgian and Spanish, where the MS2S and ST-Base models were better, respectively, although not with a significant difference. On the test set, the MS2S model gave better results only for Georgian and Hungarian.

[3] `https://github.com/clab/cnn`

Table 1: Results for inflection generation (first sub-task), measuring accuracy on the development set: our models vs. the shared task (ST-Base) and Factored (Fact.) baselines, and mean reciprocal rank (MRR) on the test set: our models vs. the best performing model (Kann and Schütze, 2016).

| | Dev | | | | Test | | |
|---|---|---|---|---|---|---|---|
| Language | ST-Base | Fact. | MS2S | NDST | MS2S | NDST | Best |
| Russian | 90.38 | 84.22 | 91.57 | **93.33** | 89.73 | 90.62 | **91.46** |
| Georgian | 89.83 | 92.37 | **98.41** | 97.01 | 97.55 | 96.54 | **98.5** |
| Finnish | 68.27 | 75.78 | **95.8** | 94.36 | 93.81 | 92.58 | **96.8** |
| Arabic | 70.29 | 24.04 | **96.28** | 92.95 | 93.34 | 89.96 | **95.47** |
| Navajo | 71.9 | 83.47 | **98.82** | 98.48 | 80.13 | 88.43 | **91.48** |
| Spanish | 96.92 | 91.79 | 98.99 | **99.31** | 98.41 | 98.33 | **98.84** |
| Turkish | 59.17 | 64.68 | **98.18** | 97.8 | 97.74 | 96.17 | **98.93** |
| German | 89.29 | 90.35 | **96.36** | 95.99 | 95.11 | 94.87 | **95.8** |
| Hungarian | 78.62 | 65.75 | **99.23** | 98.76 | 98.33 | 97.59 | **99.3** |
| Maltese | 36.94 | N/A | **87.92** | 85.2 | 82.4 | 84.78 | **88.99** |

Table 2: Results for morphological re-inflection with source attributes (second sub-task) measuring accuracy over the development set: our models vs. the shared task (ST-Base) baseline, and mean reciprocal rank (MRR) over the test set: our models vs. the best performing model (Kann and Schütze, 2016)

| | Dev | | | Test | | |
|---|---|---|---|---|---|---|
| Language | ST-Base | MS2S | NDST | MS2S | NDST | Best |
| Russian | 85.63 | 85.06 | **86.62** | 83.36 | 85.81 | **90.11** |
| Georgian | 91.5 | **94.13** | 93.81 | 92.65 | 92.27 | **98.5** |
| Finnish | 64.56 | 77.13 | **84.31** | 74.44 | 80.91 | **96.81** |
| Arabic | 58.75 | 75.25 | **78.37** | 70.26 | 73.95 | **91.09** |
| Navajo | 60.85 | 63.85 | **75.04** | 56.5 | 67.88 | **97.81** |
| Spanish | **95.63** | 93.25 | 95.37 | 92.21 | 94.26 | **98.45** |
| Turkish | 54.88 | 82.56 | **87.25** | 81.69 | 83.88 | **98.38** |
| German | 87.69 | 93.13 | **94.12** | 91.67 | 92.66 | **96.22** |
| Hungarian | 78.33 | 94.37 | **94.87** | 92.33 | 91.16 | **99.42** |
| Maltese | 26.2 | 43.29 | **49.7** | 41.92 | 50.13 | **86.88** |

Table 3: Results for morphological re-inflection without source attributes (third sub-task) measuring accuracy over the development set: our models vs. the shared task (ST-Base) baseline, and mean reciprocal rank (MRR) over the test set: our models vs. the best performing model (Kann and Schütze, 2016)

| | Dev | | | Test | |
|---|---|---|---|---|---|
| Language | ST-Base | MS2S | NDST | MS2S | Best |
| Russian | 81.31 | **84.56** | 84.25 | 82.81 | **87.13** |
| Georgian | 90.68 | **93.62** | 91.05 | 92.08 | **96.21** |
| Finnish | 61.94 | **76.5** | 66.25 | 72.99 | **93.18** |
| Arabic | 50 | **72.56** | 69.31 | 69.05 | **82.8** |
| Navajo | 60.26 | **62.7** | 54.0 | 52.85 | **83.5** |
| Spanish | 88.94 | **92.62** | 89.68 | 92.14 | **96.69** |
| Turkish | 52.19 | **79.87** | 75.25 | 79.69 | **95.0** |
| German | 81.56 | **90.93** | 89.31 | 89.58 | **92.41** |
| Hungarian | 78 | **94.25** | 83.83 | 91.91 | **98.37** |
| Maltese | 24.75 | **44.04** | 3.58 | 40.79 | **84.25** |

For the third task we also measured performance with respect to ST-Base as can be seen in Table 3. On the development set, the MS2S model outperformed the others on all languages. Since this was the situation we did not submit the NDST model for this sub-task, thus not showing test results for the NDST model on the test set.

## 6 Preliminary Analysis

An obvious trend we can see in the results is the MS2S approach giving higher accuracy scores on the first and third tasks, while the NDST approach being significantly better on the second task. While inspecting the data for the second and third tasks we noticed that the datasets only differ in the added morpho-syntactic attributes for the input sequences, and are identical other then that. This is encouraging as it shows how the NDST control mechanism can facilitate the additional data on the input sequence to predict inflections in a better way. We plan to further analyze the results to better understand the cases where the NDST architecture provides added value over the MS2S approach.

## 7 Discussion and Future Work

Our systems reached the second/third place in the Restricted category in the shared task, depending on the language/sub-task combination. It is also encouraging to see that if we submitted our systems as-is to the Standard and Bonus tracks we would also get similar rankings, even without using the additional training data available there. The winning submission in all tracks, described in (Kann and Schütze, 2016) also used an encoder-decoder approach that incorporated the morpho-syntactic attributes as inputs to the network, but with several differences from our approach like using an attention mechanism similar to (Bahdanau et al., 2014), training a single model for all inflection types rather than one per POS type and performing prediction by using an ensemble of five models with majority voting rather than using a single trained model like we did. Future work may include exploring a hybrid approach that combines the ideas proposed in our work and the latter. Other recent works that propose ideas relevant to explore in future work in this direction are (Gu et al., 2016), which describe a different copying mechanism for encoder-decoder architectures, or (Rastogi et al., 2016), which models the reinflection task using a finite state transducer weighted with neural context that also takes special care of the character copying issue.

## Acknowledgments

We thank Manaal Faruqui for sharing his code with us. This work was supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

## References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In Rada Mihalcea, Joyce Yue Chai, and Anoop Sarkar, editors, *HLT-NAACL*, pages 1024–1029. The Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

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 2016 Meeting of SIGMORPHON*, Berlin, Germany, August. The Association for Computational Linguistics.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *EMNLP*, pages 616–627. The Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195, Atlanta, Georgia, June. The Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, USA, June 12 - June 17, 2016.*

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393.*

Mans Hulden, Markus Forsberg, and Malin Ahlberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In Gosse Bouma and Yannick Parmentier 0001, editors, *EACL*, pages 569–578. The Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*, Berlin, Germany, August. The Association for Computational Linguistics.

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Kimmo Koskenniemi. 1983. Two-level morphology: A general computational model of word-form recognition and production. Technical Report Publication No. 11, Department of General Linguistics, University of Helsinki.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 922–931, Denver, Colorado, May–June. The Association for Computational Linguistics.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proc. of NAACL*.

Katsuhito Sudoh, Shinsuke Mori, and Masaaki Nagata. 2013. Noise-aware character alignment for bootstrapping statistical machine transliteration from bilingual corpora. In *EMNLP*, pages 204–209. The Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *NIPS*, pages 3104–3112.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *ACL*. The Association for Computational Linguistics.