Imitation Learning for Neural Morphological String Transduction

Peter Makarov **Simon Clematide**
Institute of Computational Linguistics
University of Zurich, Switzerland
makarov@cl.uzh.ch simon.clematide@cl.uzh.ch

Abstract

We employ imitation learning to train a neural transition-based string transducer for morphological tasks such as inflection generation and lemmatization. Previous approaches to training this type of model either rely on an external character aligner for the production of gold action sequences, which results in a suboptimal model due to the unwarranted dependence on a single gold action sequence despite spurious ambiguity, or require warm starting with an MLE model. Our approach only requires a simple expert policy, eliminating the need for a character aligner or warm start. It also addresses familiar MLE training biases and leads to strong and state-of-the-art performance on several benchmarks.

1 Introduction

Recently, morphological tasks such as inflection generation and lemmatization (Figure 1) have been successfully tackled with neural transition-based models over edit actions (Aharoni and Goldberg, 2017; Robertson and Goldwater, 2018; Makarov and Clematide, 2018; Cotterell et al., 2017b). The model, introduced in Aharoni and Goldberg (2017), uses familiar inductive biases about morphological string transduction such as conditioning on a single input character and monotonic character-to-character alignment. Due to this, the model achieves lower time complexity (compared to soft-attentional seq2seq models) and strong performance on several datasets.

Aharoni and Goldberg train the model by maximizing the conditional log-likelihood (MLE) of gold edit actions derived by an independent character-pair aligner. The MLE training procedure is therefore a pipeline, and the aligner is completely uninformed of the end task. This results in error propagation and the unwarranted dependence of the transducer on a single gold action sequence—in contrast to weighted finite-state transducers (WFST) that take into account all permitted action sequences. Although these problems—as well as the exposure bias and the loss-metric mismatch arising from this MLE training (Wiseman and Rush, 2016)—can be addressed by reinforcement learning-style methods (Ranzato et al., 2016; Bahdanau et al., 2017; Shen et al., 2016, RL), for an effective performance, all these approaches require warm-start initialization with an MLE-pretrained model. Another shortcoming of the RL-style methods is delayed punishment: For many NLP problems, including morphological string transduction, one can pinpoint actions that adversely affect the global score. For example, it is easy to tell if inserting some character c at step t would render the entire output incorrect. Assigning individual blame to single actions directly—as opposed to scoring the entire sequence via a sequence-level objective—simplifies the learning problem.

하다 {V, PAST, FORMAL POLITE, INTERROGATIVE, HONORIFIC}	▷	하셨습니까
하셨습니까	▷	하다

Figure 1: Morphological tasks with examples in Korean: inflection generation (top) and lemmatization (bottom). (McCune-Reischauer: 하다=hada, 하셨습니까=hasyŏssŭmnikka).

Faced with problems similar to those arising in transition-based dependency parsing with static oracles (Goldberg and Nivre, 2012), we train this model in the imitation learning (IL) framework (Daumé III et al., 2009; Ross et al., 2011; Chang et al., 2015), using a simple expert policy. Our approach eliminates both all dependency on an external character aligner and the need for MLE pre-training. By making use of exploration of past and future actions and having a global objective, it addresses the MLE training biases, while providing relevant action-level training signal. The approach leads to strong and state-of-the-art results on a number of morphological datasets, outperforming models trained with minimum risk training (MRT).

2 Model Description

We use a variant of the seq2seq state-transition system by Aharoni and Goldberg (2017). The model transduces the input string into the output string by performing single-character edits (insertions, deletions). The encoder RNN computes context-enriched representations of input characters, which are pushed onto the buffer at the beginning of transduction. The decoder RNN keeps track of the history of edits. Transitions—edits—are scored based on the output of the decoder state and can write a character or pop the representation of a character from the top of the buffer. We choose the model variant of Makarov and Clematide (2018), who add the copy edit, which results in strong performance gains in low-resource settings.

Let $\mathbf{x} = x_1 \dots x_n$, $x_i \in \Sigma_x$ be an input sequence, $\mathbf{y} = y_1 \dots y_p$, $y_j \in \Sigma_y$ an output sequence, and $\mathbf{a} = a_1 \dots a_m$, $a_t \in \Sigma_a$ an action sequence. Let $\{f_h\}_{h=1}^{H}$ be the set of all features. The morpho-syntactic description of a transduction is then an n-hot vector $\mathbf{e} \in \{0,1\}^H$.

The model employs a bidirectional long short-term memory (LSTM) encoder (Graves and Schmidhuber, 2005) to produce representations for each character of the input $\mathbf{x}$:

$$\mathbf{h}_1, \dots, \mathbf{h}_n = \text{BiLSTM}(E(x_1), \dots, E(x_n)), \quad (1)$$

where E returns the embedding for x_i. We push $\mathbf{h}_1, \dots, \mathbf{h}_n$ in reversed order onto the buffer. The transduction begins with the full buffer and the empty decoder state.

Transitions are scored based on the output of the LSTM decoder state (Hochreiter and Schmidhuber, 1997):

$$\mathbf{s}_t = \text{LSTM}(\mathbf{c}_{t-1}, [A(a_{t-1})\ ;\ \mathbf{h}_i]), \quad (2)$$

where $\mathbf{c}_{t-1}$ is the previous decoder state, $A(a_{t-1})$ is the embedding of the previous edit action, and $\mathbf{h}_i$ is the input character representation at the top of the buffer. If features are part of the input in the task, then the input to the decoder also contains the representation of morpho-syntactic description $\mathbf{e}$, $[F(f_1)\,; \dots ; F(f_H)]$, which is a concatenation of the embedded features and a designated embedding $F(0)$ is used instead of $F(f_h)$ if $e_h = 0$.

The probabilities of transitions are computed with a softmax classifier:

$$P(a_t = k \mid \mathbf{a}_{<t}, \mathbf{x}, \Theta) = \text{softmax}_k(\mathbf{W} \cdot \mathbf{s}_t + \mathbf{b}) \quad (3)$$

Model parameters Θ include $\mathbf{W}$, $\mathbf{b}$, the embeddings, and the parameters of the LSTMs.

The alphabet of edit actions Σ_a contains INSERT(c) for each $c \in \Sigma_y$, DELETE, and COPY. An INSERT(c) action outputs c; DELETE pops $\mathbf{h}_i$ from the top of the buffer; COPY pops $\mathbf{h}_i$ from the top of the buffer and outputs x_i. The system exhibits spurious ambiguity: Multiple action sequences lead to the same output string.

2.1 MLE Training

Aharoni and Goldberg train their model by minimizing the negative conditional log-likelihood of the data $D = \{(\mathbf{x}^{(l)}, \mathbf{a}^{(l)})\}_{l=1}^{N}$:

$$\mathcal{L}(D, \Theta) = -\sum_{l=1}^{N} \sum_{t=1}^{m} \log P(a_t^{(l)} \mid \mathbf{a}_{<t}^{(l)}, \mathbf{x}^{(l)}, \Theta), \quad (4)$$

where gold action sequences $\mathbf{a}^{(l)}$ are deterministically computed from a character-pair alignment of the input and output sequences $(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})$. The character-pair aligner is trained separately to optimize the likelihood of the actual training data $T = \{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$. For the details, we refer the reader to Aharoni and Goldberg (2017).

3 IL Training

One problem with the MLE approach is that the aligner is trained in a disconnect from the end task. As a result, alignment errors lead to the learning of a suboptimal transducer. Switching to a different aligner can dramatically improve performance (Makarov and Clematide, 2018). More fundamentally, in the face of the vast spurious ambiguity, the transducer is forced to adhere to a single gold action sequence whereas typically, legitimate and equally likely alternative edit sequences exist. This uncertainty is not accessible to the transducer, but could be profitably leveraged by it.

We address this problem within the IL framework and train the model to imitate an expert policy (dynamic oracle), which is a map—on the training data—from configurations to sets of optimal actions. Actions are optimal if they lead to the lowest sequence-level loss, under the assumption that all future actions are also optimal (Daumé III et al., 2009). In the *roll-in* stage, we run the model on a training sample and follow actions either returned by the expert policy (as in teacher forcing) or sampled from the model (which itself is a stochastic policy). In this way, we obtain a sequence of configurations summarized as decoder outputs $\mathbf{s}_1, \ldots, \mathbf{s}_m$. In the *roll-out* stage, we compute the sequence-level loss for every valid action a in each configuration $\mathbf{s}_t$. To this end, we execute a and then either query the expert to obtain the loss for the optimal action sequence following a or run the model for the rest of the input and evaluate the loss of the resulting action sequence. Finally, the sequence-level losses obtained in this way for all actions a enter the action-level loss for configuration $\mathbf{s}_t$ that we minimize with respect to Θ.

Sequence-level loss We define the loss in terms of the *Levenshtein distance* (Levenshtein, 1966) between the prediction and the target and the *edit cost* of the action sequence. Given input $\mathbf{x}^{(l)}$ with target $\mathbf{y}^{(l)}$, the loss from producing an action sequence $\mathbf{a}$ is:

$$\ell(\mathbf{a}, \mathbf{x}^{(l)}, \mathbf{y}^{(l)}) = \beta \, \text{distance}(\mathbf{y}, \mathbf{y}^{(l)}) + \text{cost}(\mathbf{a}), \quad (5)$$

where $\mathbf{y}$ is computed from $\mathbf{a}$ and $\mathbf{x}^{(l)}$ and $\beta \geq 1$ is some penalty for unit distance.[1] The first term represents the task objective. The second term enforces that the task objective is reached with a minimum number of edits.

The second term is crucial as it takes over the role of the character aligner. Initially, we also experimented with only Levenshtein distance as loss, similar to previous work on character-level problems (Leblond et al., 2018; Bahdanau et al., 2017). However, models did not learn much, which we attribute to sparse training signal as all action sequences producing the same $\mathbf{y}$ would incur the same sequence-level loss, including intuitively very wasteful ones, e.g. first deleting all of $\mathbf{x}^{(l)}$ and then inserting of all of $\mathbf{y}^{(l)}$.

Expert The expert policy keeps track of the prefix of the target $\mathbf{y}^{(l)}$ in the predicted sequence $\mathbf{y}_{<t}$ and returns actions that lead to the completion of the suffix of $\mathbf{y}^{(l)}$ using an action sequence with the lowest edit cost. The resulting prediction $\mathbf{y}$ attains the minimum edit distance from $\mathbf{y}^{(l)}$. For example, if $\mathbf{x}^{(l)} = walk$ and $\mathbf{y}^{(l)} = walked$, the top of the buffer is $\mathbf{h}_3$ representing $x_3 = l$, and $\mathbf{y}_{<3} = wad$ due to a sampling error from a roll-in with the model, the expert returns $\{\text{COPY}\}$.

[1]We use unit costs to compute edit cost and distance.

Action-level loss Given sequence-level losses, we compute the regret for each action a:

$$r_t(a) = \ell(\mathbf{a}, \mathbf{x}^{(l)}, \mathbf{y}^{(l)}) - \min_{a' \in A(\mathbf{s}_t)} \ell(\mathbf{a}', \mathbf{x}^{(l)}, \mathbf{y}^{(l)}), \quad (6)$$

where $\mathbf{a}$ (or $\mathbf{a}'$) is the action sequence resulting from taking a (or a') at $\mathbf{s}_t$ and $A(\mathbf{s}_t)$ is the set of valid actions. Thus, $r_t(a)$, which quantifies how much we suffer from taking action a relative to the optimal action under the current policy, constitutes the direct blame of a in the sequence-level loss.

Classic IL employs cost-sensitive classification, with regrets making up costs (Daumé III et al., 2009; Chang et al., 2015). Our initial experiments with cost-sensitive classification resulted in rather inefficient training and not very effective models. Instead, we choose to minimize the negative marginal log-likelihood of all optimal actions (Riezler et al., 2000; Goldberg, 2013; Ballesteros et al., 2016). Given the training data $T = \{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$, the action-level loss is:

$$\mathcal{L}(T, \Theta) = -\sum_{l=1}^{N} \sum_{t=1}^{m} \log \sum_{a \in A_t} P(a \,|\, \mathbf{a}_{<t}, \mathbf{x}^{(l)}, \Theta), \quad (7)$$

where $A_t = \{a \in A(\mathbf{s}_t) : r_t(a) = 0\}$, the set of optimal actions under the current policy. Depending on the roll-in schedule, the next edit a_{t+1} is sampled either uniformly at random from A_t or from the distribution of valid edits. To include all the computed regrets into the loss, we also experiment with the cost-augmented version of this objective (Gimpel and Smith, 2010), where regrets function as costs.

The downside of IL is that roll-outs are costly. We avoid computing most of the roll-outs by checking if an action increases the edit distance from $\mathbf{y}^{(l)}$. If it does, we heuristically assign this action a regret of β. We use this heuristic in both expert and model roll-outs.

4 Experiments

We demonstrate the effectiveness of our approach on three tasks: inflection generation (using the typologically diverse SIGMORPHON 2016 and SIGMORPHON 2017 datasets of Cotterell et al. (2016, 2017a)), reinflection (the small-sized German CELEX dataset of Dreyer et al. (2008)), and

Model	RU	DE	ES	KA	FI	TR	HU	NV	AR	MT	Avg.
MED	91.5	95.8	98.8	98.5	95.5	98.9	96.8	91.5	**99.3**	89.0	95.6
SOFT	92.2	96.5	98.9	**98.9**	97.0	**99.4**	97.0	**95.4**	**99.3**	88.9	**96.3**
HA	92.2	96.6	98.9	98.1	95.9	98.0	96.2	93.0	98.8	88.3	95.6
HA*	92.0	96.3	98.9	97.9	95.8	97.6	98.8	92.1	95.1	87.8	95.2
CA	91.9	96.4	98.8	98.3	96.5	97.7	98.9	92.1	94.6	87.7	95.3
CA-D	**92.4**	**96.6**	**98.9**	98.7	97.2	98.5	99.3	95.2	96.5	**89.2**	96.2
CA-R	92.3	96.5	98.9	**98.9**	**97.3**	98.9	**99.4**	95.2	96.1	88.8	96.2

Table 1: Results on SIGMORPHON 2016 data.[2]

Model	L	M
SGM17TOP	50.6	82.8
HA*	31.5	80.2
CA	48.8	81.0
HA*-MRT	33.1	81.5
CA-MRT	49.9	82.9
CA-MRT-A	49.9	82.7
CA-D	50.3	82.6
CA-R	**51.6**	83.8
CA-RM	50.6	**84.0**

Table 4: Results on Low and Medium settings of SIGMORPHON 2017 data (averaged over 52 languages).

-MRT: minimum risk training; **-MRT-A**: MRT with action cost in the loss; **-D**: only expert roll-outs; **-R**: expert and model roll-outs; **-RM**: softmax-margin, expert and model roll-outs

Model	13SIA	2PIE	2PKE	rP	Avg.
LAT	**87.5**	93.4	87.4	84.9	88.3
NWFST	85.1	94.4	85.5	83.0	87.0
HA*	84.6	93.9	88.1	85.1	87.9
CA	85.0	94.5	88.0	84.9	88.1
HA*-MRT	84.8	94.0	88.1	85.2	88.0
CA-MRT	85.6	**94.6**	88.0	85.3	88.4
CA-D	85.7	94.4	**88.4**	85.1	88.4
CA-R	85.6	94.4	88.3	**85.3**	**88.4**
CA-RM	84.9	94.1	88.3	85.0	88.1

Table 2: Results on CELEX data.

Model	EU	EN	GA	TL	Avg.
LAT	93.6	96.9	97.9	88.6	94.2
NWFST	91.5	94.5	97.9	97.4	95.3
LEM[3]	96.5	96.3	**98.7**	**98.8**	97.6
HA*	**97.0**	**97.5**	97.9	98.3	**97.7**
CA	96.3	96.9	97.7	98.3	97.3
CA-D	96.1	97.0	97.7	98.4	97.3
CA-R	96.6	97.2	97.5	98.3	97.4
CA-RM	96.5	97.0	97.8	98.3	97.4

Table 3: Lemmatization results.

Experimental results. Soft-attention seq2seq models: MED=Kann and Schütze (2016) (cited from Aharoni and Goldberg (2017)), SOFT=Aharoni and Goldberg (2017), LEM=Bergmanis and Goldwater (2018). WFSTs: LAT=Dreyer et al. (2008), NWFST=Rastogi et al. (2016). Transition-based models: HA=Aharoni and Goldberg (2017), SGM17TOP=Makarov et al. (2017), and from Makarov and Clematide (2018): HA*=reimplementation of HA, CA=model in §2, and HA*-MRT, CA-MRT (risk=normalized edit distance). We report exact-match accuracies for ensembles of 5 models (SIGM. 2016 and 2017) and single-model averages over 5 folds (CELEX) and 10 folds (lemmatization).

lemmatization (the standard subset of the Wicentowski (2002) dataset).

We use character and feature embeddings of size 100 and 20, respectively, and one-layer LSTMs with hidden-state size 200. Following Aharoni and Goldberg, for every character $c \in \Sigma_x \cap \Sigma_y$, we let $A(\text{INSERT}(c)) := E(c)$, i.e. the same embedding represents both c and the insertion of c. We optimize with ADADELTA (Zeiler, 2012), use early stopping and batches of size 1. We set the penalty for unit distance $\beta = 5$ and roll in with an inverse-sigmoid decay schedule as in Bengio et al. (2015). CA-D models are trained with expert roll-outs only (as is often the case in dynamic-oracle parsing). CA-R and CA-RM models mix expert and learned roll-outs with probability 0.5 as in Chang et al. (2015). CA-RM models optimize softmax-margin.

For comparison, we also train models with MRT (CA-MRT-A) as in Shen et al. (2016), using a global objective similar to our sequence-level loss (Eq. 5). We use batches of at most 20 unique samples per training example. The risk is a convex combination of normalized Levenshtein distance and the action sequence cost, which we min-max scale, within a batch, to the [0, 1] interval.

We decode all our models using beam search with beam width 4.

Our approach performs best on most languages of the SIGMORPHON 2016 data (Table 1) and both limited-resource settings of SIGMORPHON 2017 (Table 4). It achieves marginal improvement over an MRT model on the reinflection task (Table 2) with consistent gains on the 2PKE$\mapsto$z transformation (Dreyer et al., 2008), that involves infixation. Using mixed roll-outs (CA-R, CA-RM) improves performance on the SIGMORPHON 2017 inflection data (Table 4), otherwise the results are close to CA-D. We also note strong gains over CA-MRT-A trained with a similar global loss (Table 4). Generally, improvements are most pronounced in inflection generation, the only task where the model could profit from adjusting alignment to available feature information (cf. Table 3).

[2]Language codes: RU=Russian, DE=German, ES=Spanish, KA=Georgian, FI=Finnish, TR=Turkish, HU=Hungarian, NV=Navajo, AR=Arabic, MT=Maltese, EU=Basque, EN=English, GA=Irish, TL=Tagalog.

[3]Personal communication.

We take a closer look at the results in the SIGMORPHON 2017 medium data-size setting (1,000 training examples per language). CA-RM makes the largest performance gains on languages with complex morphological phenomena (Semitic and Uralic languages, Navajo) and an above average number of unique morpho-syntactic descriptions. Khaling and Basque, outliers with 367 and 740 unique morpho-syntactic descriptions in the training data, are among the top five languages with the largest gains. The lowest gains and rare losses are made for Romance and Germanic languages and languages with many unique morpho-syntactic descriptions but regular morphologies (Quechua, Urdu/Hindi).

5 Related work

Traditionally, morphological string transduction has been approached with discriminative weighted finite-state transducers (Rastogi et al., 2016; Cotterell et al., 2014; Dreyer et al., 2008; Eisner, 2002). Yu et al. (2016) and Graves (2012) tackle the modeling of unbounded dependencies in the output, while preserving latent monotonic hard character alignment. Faruqui et al. (2016); Kann and Schütze (2016) successfully apply seq2seq modeling to the task. Aharoni and Goldberg (2017) introduce a neural version of the transition-based model over edits. Makarov and Clematide (2018) show gains from using the copy edit and address the MLE training biases with MRT.

The limitations of teacher forcing have recently been the focus of intense research (Edunov et al., 2017; Wiseman and Rush, 2016; Shen et al., 2016), including the adaptation of RL methods (Ranzato et al., 2016; Bahdanau et al., 2017). Most of these approaches require warm start with an MLE model and themselves introduce discrepancies between training with sampling and search-based decoding. Such biases do not arise from IL, which has recently been proposed for seq2seq models (Leblond et al., 2018). Our approach, related to Leblond et al. (2018), additionally addresses the problem of spurious ambiguity, which is not present in seq2seq models.

6 Conclusion

We show that training to imitate a simple expert policy results in an effective neural transition-based model for morphological string transduction. The fully end-to-end approach addresses various shortcomings of previous training regimes (the need for an external character aligner, warm-start initialization, and MLE training biases), and leads to strong empirical results. We make our code and predictions publicly available.[4]

Acknowledgments

We thank Mathias Müller for help with the manuscript, Toms Bergmanis for sharing with us the results of his system, and the reviewers for interesting and helpful comments. We also thank the anonymous COLING reviewer who suggested that we should look at spurious ambiguity. Peter Makarov has been supported by European Research Council Grant No. 338875, Simon Clematide by the Swiss National Science Foundation under Grant No. CRSII5_173719.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A Smith. 2016. Training with exploration improves a greedy stack-LSTM parser. In *EMNLP*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with Lematus. In *NAACL*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume III, and John Langford. 2015. Learning to search better than your teacher. In *ICML*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017a. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*.

[4]https://github.com/ZurichNLP/emnlp2018-imitation-learning-for-neural-morphology

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*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *ACL*.

Ryan Cotterell, John Sylak-Glassman, and Christo Kirov. 2017b. Neural graphical models over strings for principal parts morphological paradigm completion. In *EACL*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *ACL*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. In *NAACL*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *ACL*.

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

Kevin Gimpel and Noah A Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *NAACL-HLT*.

Yoav Goldberg. 2013. Dynamic-oracle transition-based parsing with calibrated probabilistic output. In *IWPT*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING*.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

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

Rémi Leblond, Jean-Baptiste Alayrac, Anton Osokin, and Simon Lacoste-Julien. 2018. SEARNN: Training RNNs with global-local losses. In *ICLR*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10.

Peter Makarov and Simon Clematide. 2018. Neural transition-based string transduction for limited-resource setting in morphology. In *COLING*.

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*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

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

Stefan Riezler, Jonas Kuhn, Detlef Prescher, and Mark Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *ACL*.

Alexander Robertson and Sharon Goldwater. 2018. Evaluating historical text normalization systems: How well do they generalize? In *NAACL*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *ACL*.

Richard Wicentowski. 2002. *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*. Ph.D. thesis, Johns Hopkins University.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *EMNLP*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. arXiv:1212.5701.