IT–IST at the SIGMORPHON 2019 Shared Task: Sparse Two-headed Models for Inflection

Ben Peters[†] and **André F.T. Martins**[†‡]
[†]Instituto de Telecomunicações, Lisbon, Portugal
[‡]Unbabel, Lisbon, Portugal
benzurdopeters@gmail.com, andre.martins@unbabel.com

Abstract

This paper presents the Instituto de Telecomunicações–Instituto Superior Técnico submission to Task 1 of the SIGMORPHON 2019 Shared Task. Our models combine sparse sequence-to-sequence models with a two-headed attention mechanism that learns separate attention distributions for the lemma and inflectional tags. Among submissions to Task 1, our models rank second and third. Despite the low data setting of the task (only 100 in-language training examples), they learn plausible inflection patterns and often concentrate all probability mass into a small set of hypotheses, making beam search exact.

1 Introduction

Morphological inflection is the task of producing an inflected form, given a lemma and a set of inflectional tags. A widespread approach to the task is the attention-based sequence-to-sequence model (*seq2seq*; Bahdanau et al., 2015; Kann and Schütze, 2016); such models perform well but are difficult to interpret. To mitigate this shortcoming, we employ an alternative architecture which combines sparse *seq2seq* modeling (Peters et al., 2019) with two-headed attention that attends separately to the lemma and inflectional tags (Ács, 2018). The attention and output distributions are computed with the sparsemax function and models are trained to minimize sparsemax loss (Martins and Astudillo, 2016). Sparsemax, unlike softmax, can assign exactly zero attention weight to irrelevant source tokens and exactly zero probability to implausible hypotheses. We apply our models to Task 1 at the SIGMORPHON 2019 Shared Task (McCarthy et al., 2019), which extends morphological inflection to a cross-lingual setting. We present two sparse *seq2seq* architectures:

- DOUBLEATTN (`it-ist-01-1`) is a reimplementation of the two-headed attention model (Ács, 2018) which substitutes sparsemax and its loss for softmax and cross entropy loss. It uses separate encoders and attention heads for the lemma and inflections, and concatenates the outputs of the attention heads.
- GATEDATTN (`it-ist-02-1`) replaces the attention concatenation with a sparse gate which interpolates the lemma and inflection attention. The intuition is that the lemma and inflectional tags are not likely to be equally important at all time steps. For example, in a suffixing language, the first several generated characters are likely to be identical to the lemma; inflectional tags are not relevant. The sparse gate allows the model to learn to shift focus between the two attentions while ignoring the other at a given time step.

GATEDATTN and DOUBLEATTN rank second and third, respectively, among submissions to Task 1. In addition, their behavior is highly interpretable: they mostly learn to attend to a single lemma hidden state at a time, progressing monotonically from left to right, while their inflection attention learns patterns which reflect underlying morphological structure. The sparse output layer often allows the model to concentrate all probability mass into a single hypothesis, providing a certificate that decoding is exact. Our analysis shows that sparsity is also highly predictive of performance on the shared task metrics, showing that the models "know what they know".

2 Models

Our architecture is mostly the same as a standard RNN-based *seq2seq* model with attention. In this section, we outline the changes needed to extend this model to use sparsemax and two-headed attention in a multilingual setting.

Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 50–56
Florence, Italy. August 2, 2019

2.1 Sparsemax

Our models' sparsity comes from the sparsemax function (Martins and Astudillo, 2016), which computes the Euclidean projection of a vector $\boldsymbol{z} \in \mathbb{R}^n$ onto the n–dimensional probability simplex $\triangle^n \coloneqq \{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p} \geq 0, \mathbf{1}^\top \boldsymbol{p} = 1\}$:

$$\mathsf{sparsemax}(\boldsymbol{z}) \coloneqq \operatorname*{argmin}_{\boldsymbol{p} \in \triangle^n} \|\boldsymbol{p} - \boldsymbol{z}\|^2 \qquad (1)$$

Like softmax, sparsemax converts an arbitrary real-valued vector into a probability distribution. The critical difference is that sparsemax can assign exactly zero probability, whereas softmax is strictly positive. Sparsemax is differentiable almost everywhere and can be computed quickly, allowing its use as a drop-in replacement for softmax. It has previously been used in *seq2seq* for computing both attention weights (Malaviya et al., 2018) and output probabilities (Peters et al., 2019). Sparse attention is attractive in morphological inflection because it resembles hard attention, which has been successful on the task (Aharoni and Goldberg, 2017; Wu et al., 2018).

2.2 Encoder–Decoder Model

Multilingual embeddings Each encoder and decoder uses an embedding layer to convert one-hot token representations into dense embeddings. To account for the bilingual nature of Task 1, each of our embedding layers contains two look-up tables: one for the sequence of input tokens, and the other for the language of the sequence. At each time step, the current token's embedding is concatenated to a language embedding.[1] Each encoder and decoder uses a separate embedding layer; no weights are tied. Characters and inflectional tags use embeddings of size D_c, while language embeddings are of size D_ℓ. Thus the total embedding size is $D = D_c + D_\ell$.

Encoders The lemma and inflection encoders are both bidirectional LSTMs (Graves and Schmidhuber, 2005). An encoder's forward and backward hidden states are concatenated, forming a sequence of source hidden states. We set the size of these hidden states as D in all experiments.

Decoder The decoder is a unidirectional LSTM (Hochreiter and Schmidhuber, 1997) with input feeding (Luong et al., 2015). At time step t, it computes a hidden state $\boldsymbol{s}_t \in \mathbb{R}^D$. Conditioned on $\boldsymbol{s}_t$ and the hidden state sequences from the lemma and inflection encoders, a two-headed attention mechanism then computes an attentional hidden state $\tilde{\boldsymbol{s}}_t \in \mathbb{R}^D$. The decoder LSTM is initialized only with the lemma encoder's state.

Attention head At time t, an attention head computes a context vector $\boldsymbol{c}_t \in \mathbb{R}^D$ conditioned on the decoder state $\boldsymbol{s}_t$ and an encoder state sequence $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_J]$. A head consists of two modular components:

1. **Alignment** Compute a vector $\boldsymbol{a} \in \mathbb{R}^J$ of alignment scores between $\boldsymbol{s}_t$ and $\boldsymbol{H}$. We use the general attention scorer (Luong et al., 2015), which computes $a_j \coloneqq \boldsymbol{s}_t^\top \boldsymbol{W}_a \boldsymbol{h}_j$.

2. **Context** Compute the context vector $\boldsymbol{c}_t$ as a weighted sum of $\boldsymbol{H}$: $\boldsymbol{c}_t \coloneqq \sum_{j=1}^J \pi_j \boldsymbol{h}_j$, where $\boldsymbol{\pi} = \mathsf{sparsemax}(\boldsymbol{a})$ is a sparse vector of alignment scores in the simplex.

In Luong et al.'s single-headed attention, the attentional hidden state $\tilde{\boldsymbol{s}}_t = \tanh(\boldsymbol{W}_s[\boldsymbol{c}_t; \boldsymbol{s}_t])$ is computed by a concatenation layer from the context vector and pre-attention hidden state. However, our two-headed attention mechanism produces two context vectors and so must be computed differently. We use two different formulations, which we describe next.

DOUBLEATTN uses the same strategy for combining multiple context vectors as Ács (2018): the lemma and inflection context vectors $\boldsymbol{u}_t$ and $\boldsymbol{v}_t$ and the target hidden state $\boldsymbol{s}_t$ are inputs to a concatenation layer:

$$\tilde{\boldsymbol{s}}_t = \mathsf{tanh}(\boldsymbol{W}_d[\boldsymbol{u}_t; \boldsymbol{v}_t; \boldsymbol{s}_t]) \qquad (2)$$

where $\boldsymbol{W}_d \in \mathbb{R}^{D \times 3D}$.

GATEDATTN, on the other hand, computes separate candidate attentional hidden states for the two context vectors:

$$\tilde{\boldsymbol{s}}_{ut} = \mathsf{tanh}(\boldsymbol{W}_u[\boldsymbol{u}_t; \boldsymbol{s}_t]) \qquad (3)$$
$$\tilde{\boldsymbol{s}}_{vt} = \mathsf{tanh}(\boldsymbol{W}_v[\boldsymbol{v}_t; \boldsymbol{s}_t]) \qquad (4)$$

where $\boldsymbol{W}_u, \boldsymbol{W}_v \in \mathbb{R}^{D \times 2D}$. We define gate weights $\boldsymbol{W}_g \in \mathbb{R}^{2 \times 3D}$ and gate bias $\boldsymbol{b}_g \in \mathbb{R}^2$ and

[1] The language embedding is the same at all time steps within an example; there is no code-switching in this task.

Model	Acc. ↑	Lev. Dist. ↓
DOUBLEATTN	48.999	1.2918
GATEDATTN	50.179	1.3209
Baseline (Wu and Cotterell, 2019)	48.549	1.3281

Table 1: Task 1 test results on the SIGMORPHON 2019 Shared Task, averaged across language pairs.

use a sparse gate to compute weights $\boldsymbol{p}_t \in \triangle^2$ for the two candidate states:

$$\boldsymbol{p}_t = \mathsf{sparsemax}(\boldsymbol{W}_g[\boldsymbol{u}_t; \boldsymbol{v}_t; \boldsymbol{s}_t] + \boldsymbol{b}_g) \quad (5)$$

We then stack the two candidate states $\tilde{\boldsymbol{s}}_{ut}$ and $\tilde{\boldsymbol{s}}_{vt}$ into a matrix $\tilde{\boldsymbol{S}}_t \in \mathbb{R}^{2\times D}$ and use the gate weights to compute $\tilde{\boldsymbol{s}}_t$ as a weighted sum of them:

$$\tilde{\boldsymbol{s}}_t = \boldsymbol{p}_t \tilde{\boldsymbol{S}} \quad (6)$$

Just as a two-dimensional softmax is equivalent to a sigmoid, this two-dimensional sparsemax is a hard sigmoid, as was pointed out by Martins and Astudillo (2016). It provides extra interpretability in the form of a three-way answer about what is relevant at a time step: the lemma, the inflections, or both.

Sparse outputs After the attentional hidden state is computed, an output layer computes scores for each output type $\boldsymbol{z} = \boldsymbol{W}_z \tilde{\boldsymbol{s}} + \boldsymbol{b}_z$. These are then converted into a sparse probability distribution $\boldsymbol{p}^\star = \mathsf{sparsemax}(z)$. The model is trained to minimize the sparsemax loss (Martins and Astudillo, 2016), defined as

$$\mathsf{L}_{\mathsf{sparsemax}}(y, \boldsymbol{z}) \coloneqq \frac{1}{2}(\|\boldsymbol{e}_y - \boldsymbol{z}\|^2 - \|\boldsymbol{p}^\star - \boldsymbol{z}\|^2) \quad (7)$$

where y is the index of the gold target and $\boldsymbol{e}_y$ is a one-hot vector. The sparsemax loss is differentiable, convex, and has a margin, and its gradient is sparse. Although softmax-based models use the cross entropy loss, this is not possible for our models because the cross entropy loss is infinite when the model assigns zero probability to the gold target.

3 Results

Our test results are shown in Table 1. Our two models ranked second and third among official submissions to Task 1.

Hyperparameter	Value
Character embedding size	180
Tag embedding size	180
Language embedding size	20
RNN size	200
Lemma encoder layers	2
Inflection encoder layers	{1, 2}
Dropout	{0.3, 0.4, 0.5}

Table 2: Hyperparameters for all models. Bracketed values were tuned individually for each language pair.

3.1 Experimental set-up

Each model was trained with early stopping for a maximum of 30 epochs with a batch size of 64. We used the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-3}, which was halved when validation accuracy failed to improve for three consecutive epochs. We tuned the dropout and the number of inflection encoder layers on a separate grid for each language pair. Our hyperparameter ranges are shown in Table 2. At test time, we decoded with a beam size of 5. We oversampled the low resource training data 100 times and did not use synthetic data or filter the corpora. We implemented our models in PyTorch (Paszke et al., 2017) with a codebase derived from OpenNMT-py (Klein et al., 2017).[2]

Hyperparameters for Uzbek The Uzbek training set contains only 1060 examples, much smaller than the other high resource corpora, and initial results with Uzbek language pairs were poor. We improved performance by oversampling the Uzbek data 10 times (yielding roughly the same high-low balance as the other pairs) and reducing the initial learning rate to 10^{-4}.

4 Analysis

Next we interpret our models' behavior on a selection of language pairs from Task 1.

4.1 Sparse Attention

Table 3 shows the sparsity of our attention mechanisms, averaged across language families. The attention is extremely sparse, especially over the lemma: models attend to fewer than 1.1 lemma characters per target character on average. Sparsity is more varied between language families in the inflection attention. This may be explained by

[2]Our code is available at https://github.com/deep-spin/SIGMORPHON2019.

	DoubleAttn		GatedAttn			
Family	Lemma	Infl.	Lemma	Infl.	Total	#Langs
Afro-Asiatic	1.11	1.55	1.14	1.27	1.62	5
Baltic	1.02	1.33	1.02	1.34	1.54	1
Celtic	1.23	1.54	1.20	1.59	2.00	12
Dravidian	1.69	1.42	1.86	1.35	1.95	1
Germanic	1.05	1.56	1.05	1.39	1.61	17
Greek	1.15	1.52	1.18	1.62	2.11	1
Indo-Iranian	1.10	1.35	1.11	1.37	1.67	7
Murrinhpatha	1.16	1.28	1.15	1.35	1.71	1
Niger-Congo	1.07	1.30	1.04	1.09	1.11	1
NW Caucasian	1.02	1.08	1.02	1.20	1.25	2
Quechua	1.24	1.19	1.09	1.23	1.42	1
Romance	1.01	1.27	1.02	1.30	1.39	10
Slavic	1.08	1.31	1.07	1.35	1.55	9
Turkic	1.06	1.09	1.08	1.15	1.35	20
Uralic	1.03	1.21	1.04	1.31	1.48	12
Overall	1.09	1.33	1.09	1.33	1.56	100

Table 3: Average number of positions with nonzero attention per target time step on the Task 1 development sets, grouped by the family of the low resource language. For DoubleAttn, this is simply averaged over all target time steps. For GatedAttn, the lemma attention nonzeros are summed only over time steps in which the gate is active for the lemma, and similarly for the inflection attention nonzeros. The 'Total' column for GatedAttn indicates the average number of nonzeros over all time steps after accounting for the gate.

typological differences between languages, which we next analyze in detail.

Turkic languages are characterized by concatenative inflections (Bickel and Nichols, 2013b) which represent individual features (monoexponence; Bickel and Nichols, 2013a). Monoexponence should allow the inflection attention to concentrate on a single tag at a time, and Table 3 confirms that Turkic inflection attention is among the sparsest for both DoubleAttn and GatedAttn models. The Azeri attention plot in Figure 1 illustrates that the inflection attention usually focuses on a single morpheme at a time, with some discrepancies at morpheme boundaries, where other tags may be relevant because of voicing assimilation rules. Furthermore, the sparse gate generally allows the model to focus on only one attention head at a time: in the Azeri example, there is only one position at which both attention heads are used. This position is the final consonant of the lemma, which appears to change because of a phonological environment created by the suffix. The shared task results suggest that sparse inflection attention is a good inductive bias for agglutinative languages: one of our models has the best test accuracy among task submissions on 11 of 20 pairs where the low resource language is Turkic and 11 of 12 pairs in the typologically similar Uralic languages.

Germanic languages present different challenges, which may explain our models' less sparse inflection attention. Often several inflections are fused into a single affix; a familiar example is the German suffix *st*, which marks a verb as present tense, second person, and singular, but has no separable parts that represent these features individually. The North Frisian plot in Figure 1 demonstrates the less sparse nature of Germanic inflection attention. Producing "wulst" from the lemma "wel" requires both a suffix and a change to the lemma, and multiple inflectional tags are attended to at several time steps. The fusional nature of the morphology means there is not a clear alignment between the inflected sequence and the tags. This in reflected in the fact that at many time steps, DoubleAttn and GatedAttn disagree about which tags to attend to. Unlike in the Turkic example, GatedAttn's gate usually gives weight to both attention heads. This makes sense because the inflection requires a change to the lemma, not

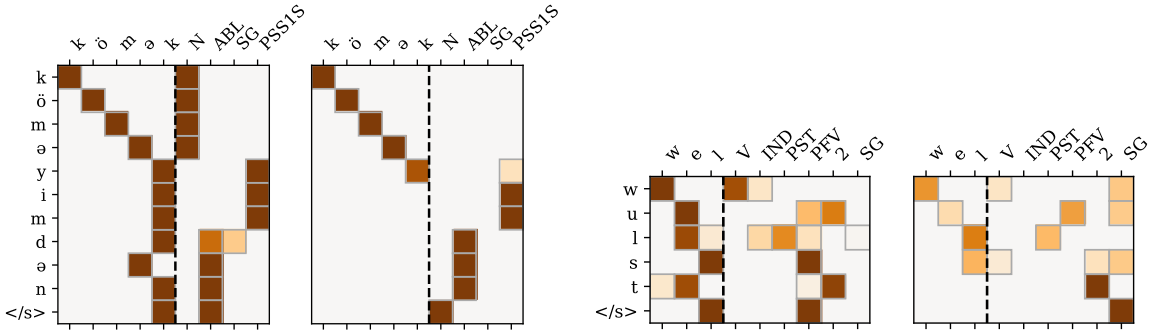

Figure 1: Pairs of attention plots for Azeri (left) and North Frisian (right) words with models trained on the Turkish–Azeri and Danish–North Frisian language pairs. Within each pair, the left plot comes from the DOUBLEATTN model and the right plot from GATEDATTN. Gray squares have zero attention weight. The dashed vertical line separates the lemma and inflection attention heads. Attention values in the GATEDATTN plots are scaled by gate weights, which is why there is no inflection attention for the first several positions of the Azeri word.

just a suffix that follows it.

4.2 Sparse Output Layer

Sparse output probabilities provide tools for analysis that are not available to softmax-based models: when no hypotheses are pruned, they provide a certificate that beam search is exact; when only one hypothesis is possible, this gives an indication of the model's certainty; and when probability is distributed among a small set of hypotheses, it is easy to reason about what phenomena continue to confuse the model.

Certainty When the probability distribution is completely concentrated at each time step, the model will be able to generate only one hypothesis, regardless of the beam width. When this happens for a particular input, the model can be said to be **certain** for that input. This also trivially guarantees that beam search is exact because no hypotheses have been pruned. As Figure 2 shows, certainty is a strong indication of performance. This suggests future work using certainty as a validation metric alternative to accuracy and loss.

Interpretable ambiguity Our Turkish–Azeri GATEDATTN model demonstrates that there is also useful information to be gleaned from the cases where the model produces multiple hypotheses. Of the 100 examples in the development set, GATEDATTN concentrated all probability mass into a single hypothesis for 79 of them, but the other 21 examples exhibit ambiguities that have linguistically plausible interpretations:

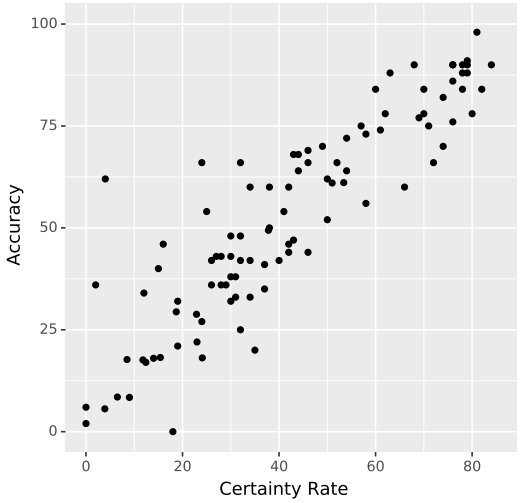

Figure 2: Percentage of inputs for which the selected GATEDATTN model concentrates all probability into a single hypothesis compared to the word-level accuracy on the development set. Each point is a language pair.

- **Consonant alternations** In 13 of the 21 examples, the hypotheses differ in their treatment of stop consonants, which have very similar phonological alternations in the two languages that are represented in orthography. The ambiguity is a sign that the model has not mastered Azeri phonological rules. Nine of the examples concern lemma-final "k" and "q", which have slightly different rules in Azeri than Turkish.[3]

[3]This judgment is based on inspection of the Azeri data and prior knowledge of Turkish.

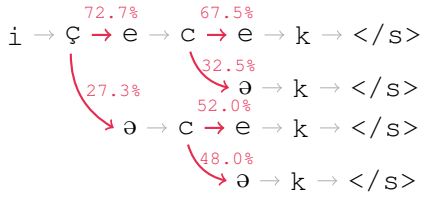

Figure 3: The Turkish–Azeri GATEDATTN model's full beam search for the Azeri lemma "içmek" and the tags V 3 SG FUT. All other sequences have zero probability. The correct form is "içəcək", while the model prefers "içecek", which would be correct with Turkish vowel harmony rules.

- **Vowel harmony** In two other examples, the model produced multiple guesses for the vowels in the future tense marker. One of these examples is shown in Figure 3. In both instances, Azeri vowel harmony rules would generate "ə" in the suffix, but the model instead produced "e", which is correct with Turkish vowel harmony. This shows the influence of the high resource language.

- **Other cases** The last six non-certain examples consist of a loanword with an unusual character sequence, two instances where one hypothesis has the wrong possessive suffix, and two where a hypothesis inserts or drops a character. The top prediction was nonetheless correct in all six.

This sort of analysis is not possible with traditional dense models because probability can never become concentrated in a small set of hypotheses and it is impossible to separate legitimate ambiguities from the long tail of implausible hypotheses.

Paradigm completion Figure 3 suggests that our models do a good job of concentrating probability in a small number of hypotheses. This raises the question of whether, by underspecifying the inflectional tags, the set of possible hypotheses in the beam can be made to resemble a lemma's complete paradigm. To investigate, we trained monolingual models with the English, German, and Turkish data from the high resource setting of Task 1 of the CoNLL–SIGMORPHON 2018 Shared Task (Cotterell et al., 2018). We used mostly the same hyperparameters as for this year's submission, except that there are no language embeddings, and the inflection tags are not used and the models have single-headed attention over the lemma sequences. We increased the beam width to 10 in order to accomodate the models' greater uncertainty. With English, this often works well: for the regular verb "jitter", the model's only possible hypotheses are "jittered", "jittering", "jitters", and "jitter", which is the complete paradigm. Irregular verbs often have a handful of other hypotheses, and sometimes the beam gives some probability to misspellings. Something similar can be seen in German, although the beam rarely contains all surface forms. For German nouns, the beam often shows uncertainty about plural formation: the hypotheses for "Nadelbaum" include "Nadelbaume", "Nadelbäume", and "Nadelbäumer", all of which are plausible German plurals. Turkish has very large paradigms, so in general it is not possible to fit all forms into a beam of any reasonable size. However, the hypotheses in the beam do typically correspond to correct forms.

5 Conclusion

We presented a new style of *seq2seq* model which brings together two-headed attention (Ács, 2018) and sparse modeling for morphological inflection (Peters et al., 2019). Our models learn sparse attention distributions in both attention heads. Their sparse probability distribution over hypotheses often allows beam search to become exact, while the remaining ambiguities often have a clear linguistic interpretation. The two versions of our model rank second and third among submissions to Task 1.

Acknowledgments

This work was supported by the European Research Council (ERC StG DeepSPIN 758969), and by the Fundação para a Ciência e Tecnologia through contracts UID/EEA/50008/2019 and CMUPERI/TIC/0046/2014 (GoLocal). We thank Gonçalo Correia, Aitor Egurtzegi, Erick Fonseca, Pedro Martins, Tsvetomila Mihaylova, Vlad Niculae, Marcos Treviso, and the anonymous reviewers, for helpful discussion and feedback.

References

Judit Ács. 2018. BME-HAS system for CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 121–126.

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

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

Balthasar Bickel and Johanna Nichols. 2013a. Exponence of selected inflectional formatives. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Balthasar Bickel and Johanna Nichols. 2013b. Fusion of selected inflectional formatives. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Gėraldine 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. *Proc. CoNLL–SIGMORPHON*.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Katharina Kann and Hinrich Schütze. 2016. Single-Model Encoder-Decoder with Explicit Morphological Representation for Reinflection. In *Proc. ACL*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. *arXiv e-prints*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

Chaitanya Malaviya, Pedro Ferreira, and André FT Martins. 2018. Sparse and constrained attention for neural machine translation. In *Proc. ACL*.

André FT Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proc. of ICML*.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sebastian Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019. The SIGMORPHON 2019 shared task: Crosslinguality and context in morphology. In *Proceedings of the 16th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology"*, Florence, Italy. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Proc. NeurIPS Autodiff Workshop*.

Ben Peters, Vlad Niculae, and André FT Martins. 2019. Sparse Sequence-to-Sequence Models. In *Proc. ACL*.

Shijie Wu and Ryan Cotterell. 2019. Exact Hard Monotonic Attention for Character-Level Transduction. *Proc. ACL*.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proc. EMNLP*.