Tü-CL at SIGMORPHON 2023: Straight-Through Gradient Estimation for Hard Attention

Leander Girrbach
University of Tübingen
leander.girrbach@uni-tuebingen.de

Abstract

This paper describes our systems participating in the 2023 SIGMORPHON Shared Task on Morphological Inflection (Goldman et al., 2023) and in the 2023 SIGMORPHON Shared Task on Interlinear Glossing. We propose methods to enrich predictions from neural models with discrete, i.e. interpretable, information. For morphological inflection, our models learn deterministic mappings from subsets of source lemma characters and morphological tags to individual target characters, which introduces interpretability. For interlinear glossing, our models learn a shallow morpheme segmentation in an unsupervised way jointly with predicting glossing lines. Estimated segmentation may be useful when no ground-truth segmentation is available. As both methods introduce discreteness into neural models, our technical contribution is to show that straight-through gradient estimators are effective to train hard attention models.

1 Introduction

This paper describes our systems participating in the SIGMORPHON–UniMorph Shared Task on Typologically Diverse and Acquisition-Inspired Morphological Inflection Generation (Goldman et al., 2023) and the SIGMORPHON 2023 Shared Task on Interlinear Glossing. For morphological inflection, we participate in part 1, and for interlinear glossing we mainly target the closed track.

Morphological Inflection is the task of predicting the correct inflected form given a lemma and set of morphological tags. An example from the Italian dataset in the shared task is

$$\text{votare (“to vote”)} \xrightarrow{\text{V;IND;FUT;NOM(1,PL)}} \text{voteremo.}$$

The organisers of the shared task provide train, validation and test splits for 26 languages. In the case of Hebrew, 2 datasets are provided. Train splits contain 10K (lemma, tags, form) triples, validation and test splits contain 1K triples.

Interlinear glossing is the task of predicting glossing lines, which is a sequence of morphological tags, including lexical translations for each token, on the sentence level given the surface text and optionally a translation. An example of interlinear glossing taken from the train portion of the Gitksan dataset in the shared task is:

(1)	*Iin*	*dip*	*gidax̱*	*guhl*	*wilt.*
	CCNJ-1.I	1PL.I	ask	what-CN	LVB-3.II

“And we asked what he did.”

The organisers of the shared task provide train, validation and test splits for 7 typologically diverse languages. Dataset sizes differ for each language. Furthermore, the shared task features a closed track, where only surface text and a translation is available for each sentence, and an open track, where canonical morpheme segmentation and POS tags are provided as additional information.

Especially when the main focus of training machine learning models is scientific discovery, even the notoriously good performance of deep neural models (Jiang et al., 2020) may not be satisfactory. Instead, models should also yield insights into what they learn about the data. However, clear and interpretable explanations are often hard to derive from models by post-hoc analysis, although many methods exist (Holzinger et al., 2020; Burkart and Huber, 2021; Rao et al., 2022). On the other hand, self-interpretable models, i.e. models whose calculations directly reveal discrete information, are generally hard to train with gradient methods and do not reach the same effectiveness as fully continuous models (Niepert et al., 2021).

Therefore, in this work we aim at narrowing the gap between inherently interpretable models and fully continuous deep sequence-to-sequence models by demonstrating the effectiveness of straight-through gradient estimators in optimising discrete intermediate representations by gradient methods.

As applications, we construct a model type for morphological inflection that shows, without ambiguity, which subset of lemma characters and tags causes the prediction of a form character. Our proposed model for interlinear glossing enriches the given surface text with shallow morpheme segmentation.

Our main contributions are: (1) We show the effectiveness of straight-through gradient estimators for learning hard attention; (2) We present a model for morphological inflection that unambiguously shows which subset of lemma characters and tags lead to the prediction of a form character; (3) We present a model that learns shallow morpheme segmentation jointly with interlinear glossing in an unsupervised fashion.

2 ST Optimization of Hard Attention

We discuss hard attention as mappings of the following form: Let $k \in \mathbb{N}$ be the number of target positions (e.g. the number of decoder positions in a encoder-decoder sequence-to-sequence model), and $\mathbf{X} \in \mathbb{R}^{n \times d}$ the matrix containing d-dimensional feature vectors of n source elements (e.g. learned embedding vectors). Each target element $\mathbf{y}_i$, $i \in \{1, \ldots, K\}$ is calculated as a sum of source element encodings, formally $\mathbf{y}_i = \sum_{j \in \rightarrow_i} \mathbf{x}_j$ where $\mathbf{x}_j$ is the jth row vector in $\mathbf{X}$ and $\rightarrow_i \subseteq \{1, \ldots, n\}$ is the set of source elements aligned to target position i. Note that a source element may be aligned to multiple target elements, i.e. appear in $\rightarrow_i$ for different i.

This mapping can be calculated by a matrix multiplication $\xi \cdot \mathbf{X} = \mathbf{Y} \in \mathbb{R}^{k \times d}$, where columns of $\xi \in \{0,1\}^{k \times n}$ are the multi-hot encodings of index sets $(\rightarrow_i)_{i \in \{1,\ldots,K\}}$. Formally, this means

$$\xi_{i,j} = \begin{cases} 1 & \text{if } j \in \rightarrow_i \\ 0 & \text{if } j \notin \rightarrow_i \end{cases}$$

We assume ξ is a sample from a underlying categorical distribution where we can compute the marginals $\hat{\xi}_{i,j}$ that specify the probability

$$\hat{\xi}_{i,j} = \Pr[j \in \rightarrow_i]$$

of j being included in $\rightarrow_i$. For example, in the case of dot-product attention, we have $\mathbf{z} \in \mathbb{R}^{k \times n}$ the matrix product of decoder states and encoder states. Then, we obtain $\hat{\xi}$ by $\mathrm{softmax}$ over rows, and ξ by sampling from the categorical distributions defined by rows of $\hat{\xi}$. At test time, argmax is used instead of sampling.

The main problem is how to side-step sampling during gradient-based optimization, because sampling is not differentiable. One solution is the so-called straight-through estimator (Bengio et al., 2013; Jang et al., 2017; Cathcart and Wandl, 2020) which means using ξ for the forward pass, i.e. when computing model outputs, but using $\hat{\xi}$ for backpropagation, i.e. when computing gradients of model parameters w.r.t. the loss.

However, gradients of $\mathbf{X}$ are affected by the discreteness of ξ as well, because $\xi_{i,j} = 0$ also means $\mathbf{x}_j$ does not receive gradients from $\mathbf{y}_i$. Therefore, when using straight-through gradient estimation, we should use $\hat{\xi}$ when computing gradients of $\mathbf{X}$. Formally, for some differentiable function f that is applied to $\mathbf{Y}$, we set

$$\frac{\partial f(\xi \cdot \mathbf{X})}{\partial \hat{\xi}} = \mathbf{X}^T \frac{\partial f(\mathbf{Y})}{\partial \mathbf{Y}}$$
$$\frac{\partial f(\xi \cdot \mathbf{X})}{\partial \mathbf{X}} = \left(\frac{\partial f(\mathbf{Y})}{\partial \mathbf{Y}}\right)^T \cdot \hat{\xi},$$

which can be implemented as

$$\mathbf{Y} = \hat{\xi} \cdot \mathbf{X} - \mathrm{sg}\left((\hat{\xi} - \xi) \cdot \mathbf{X}\right), \qquad (1)$$

where sg is the stop-gradient function (van den Oord et al., 2017) which behaves like the identity during forward pass, but has 0 partial derivatives everywhere.

3 Applications

In this section, we describe how to apply the method from Section 2 to sequence transduction (Section 3.1) and sequence segmentation (Section 3.2). We keep formulations more general than necessary for the shared tasks, because we want to highlight that the methods apply to similar problems as well.

3.1 Sequence Transduction

Sequence Transduction means transforming an input or source sequence $s_{1:n} = s_1, \ldots, s_n$ into an output or target sequence $t_{1:m} = t_1, \ldots, t_m$. Successful model types for this tasks are neural encoder-decoder networks with attention (Bahdanau et al., 2015). These models use an encoder which computes contextual source symbol representations $\mathbf{s}_1, \ldots, \mathbf{s}_n$ and a decoder which computes autoregressive target symbol representations $\mathbf{t}_1, \ldots, \mathbf{t}_m$. Entries of the attention matrix $\hat{\xi}$ are dot products[1] of source representations and target representations, normalised to a categorical distribution over source symbols for every target symbol. Output symbols are predicted from the concatenation of the respective previous autoregressive target representation with the weighted sum of source symbol representations, where weights correspond to probabilities of the respective attention distribution. In terms of interpretability, this type of model has two problems:

Soft Attention The role of soft attention (i.e. using $\hat{\xi}$ directly) with regard to explaining model predictions is not entirely understood (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019). Therefore, we want to replace soft attention with hard attention, whose interpretability is undisputed. We replace soft attention with hard attention by sampling source elements from rows of $\hat{\xi}$ during training. The sampled index sets are used to discretise $\hat{\xi}$ into ξ. We enable end-to-end training through Equation (1).

Contextual Representations Contextual symbol encodings represent information about the whole sequence, not just the encoded symbol. In deep models, it is therefore not clear what information actually is encoded (Meister et al., 2021). For this reason, we want to use non-contextual symbol embeddings for prediction and use contextual symbol encodings only for computing $\hat{\xi}$.

However, only selecting one single source symbol by hard attention and then not using any contextual information is not sufficient for successful transduction. For example, in the case of morphological inflection discussed here, predictions have to take morphological tags and surrounding characters into account when transducing a source character. Therefore, we use two attention heads computing different kinds of attention:

1. Softmax-normalised attention $\hat{\xi}_{\text{symbol}}$ to select a single symbol to transduce.

2. Sigmoid-normalised attention $\hat{\xi}_{\text{cond}}$ to select multiple symbols as conditions. In this case, the sigmoid function σ is applied to every dot-product of encoder states and decoder states individually, yielding a Bernoulli distribution for every combination. ξ_{cond} is the result of sampling from each Bernoulli distribution. At test time, we round to 0 or 1 instead of sampling to ensure deterministic predictions.

[1]There are different ways to calculate unnormalised attention scores (Luong et al., 2015; Brauwers and Frasincar, 2023), but without loss of generality we restrict the discussion to dot-product attention.

Predictions are computed from the combined context vectors, formally

$$\begin{aligned} \mathbf{Y}^{\text{symbol}} &= \xi_{\text{symbol}} \cdot \mathbf{X}_{\text{embed}} \\ \mathbf{Y}^{\text{cond}} &= \xi_{\text{cond}} \cdot \mathbf{X}_{\text{embed}} \\ p_j(\bullet \mid s_{1:n}) &= \text{MLP}([\mathbf{Y}^{\text{symbol}}_j, \mathbf{Y}^{\text{cond}}_j]) \end{aligned} \tag{2}$$

where $\bullet$ is a placeholder to indicate distributions over the target alphabet, p_j is the distribution for the jth target symbol, and $\mathbf{X}_{\text{embed}}$ is the matrix containing non-contextual source symbol embeddings.

In this formulation, the decoder is still autoregressive, but is only involved in computing attention scores, not predictions any more. Therefore, it is entirely transparent which source symbols are responsible for which predictions. Also, the condition vector is a sum of equally weighted non-contextual symbol embeddings. The only non-transparent computation are the attention scores. Formally, the model learns a mapping $\mathcal{S} \times 2^{\mathcal{S}\times\mathbb{N}} \rightarrow \mathcal{T}$ where $\mathcal{S}$ is the source alphabet, $\mathbb{N}$ are the natural numbers (to account for multiplicities of symbols), and $2^{\mathcal{S}\times\mathbb{N}}$ indicates the power set. $\mathcal{T}$ is the target alphabet. The attention mechanism selects the contextually appropriate arguments for this mapping. A more detailed description of the concrete model architecture is in Appendix B.

Of course, the increased transparency limits the expressivity of the model. One problem is that gradient signals for encoder and decoder are insufficient, because their only remaining role is to compute attention matrices. Therefore, we train sequence transduction models in a multi-task setting, using the interpretable mechanism described above together with the typical mechanism, i.e. predicting the next target symbol from decoder state and combined contextual source symbol encodings. However, we use the same attention matrices in both cases. Predictions of type one-to-many (e.g. converting a single morphological tag to a suffix consisting of multiple characters) are also problematic: For each single target symbol, a different source symbol or condition is required. Possible solutions are augmenting the target alphabet with symbol ngrams (Liu et al., 2017) or allowing for local non-autoregressive predictions (Libovický and Helcl, 2018). However, we leave exploration of such methods to future work. Finally, condition vectors $\mathbf{Y}^{\text{cond}}$ are insensitive to order due to summing being a commutative operation. This problem can be mitigated by positional encodings, but we do not observe improvements in preliminary experiments and do not explore this option here.

3.2 Sequence Segmentation

We combine hard attention with Structured Attention proposed by Kim et al. (2017). In particular, we consider the case of sequence segmentation and propose an end-to-end trainable interlinear glossing model (for the closed tack, where this information is not given) that first performs shallow morphological segmentation[2] on input words and then predicts the gloss label for each morpheme. Note that the method is also applicable to other tasks that require sequence segmentation and further processing of resulting segments, such as joint Sandhi segmentation and morphological parsing in Sanskrit (Li and Girrbach, 2022). In contrast to Kim et al., our segment encodings respect a particular sampled segmentation due to hard attention, and do not represent expected feature values.

Encoder Model Given a sentence as input to the glossing model, we first apply a character level encoder such as BiLSTM (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005), to compute contextual character representations on the sentence level. Then, we continue processing on the word level and denote a word w by its characters $w = s_1, \dots, s_n$. Each word consists of a sequence of characters that are represented by contextual features computed in the previous step. For each character at position i, we predict a Bernoulli distribution parametrised by probability $p_i^{\text{seg}} \in (0; 1)$ that indicates whether the corresponding character is the last character of a (shallow) morpheme segment in our case. We also adopt the method by Raffel et al. (2017) to add Gaussian noise to unnormalised scores during training to encourage discreteness of segmentation probabilities.

Furthermore, each word is paired with the number of morphemes in the word. According to Leipzig Glossing Conventions (Comrie et al., 2008, Rule 2), the number of morphemes in a word is given by the number of hyphen-separated labels assigned to a word. During inference, the number of labels and therefore morphemes is not given. In this case, a straightforward solution is to start a new morpheme whenever the segmentation probability exceeds a certain threshold τ. However, we found trivial solutions for τ like $\frac{1}{2}$ not to work well, while learning to predict the number of morphemes in a word from the max-pooled character representations by a MLP works well in our case. Therefore, we adopt the latter option and leave exploration of the former method to future work. In cases where we have no information about the number of segments during training, marginalising the number of segments still remains as option.

[2]Shallow morphological segmentation means only segmenting the surface string. Contrast this to canonical segmentation, which also restores a latent canonical form of present morphemes (Kann et al., 2016).

Computing Marginals Character-level segmentation scores p_i^{seg} have to be converted to the attention matrix $\hat{\xi}$ by marginalising all segmentations. For each source element s_i, marginalisation computes the marginal probability of having a morpheme boundary at s_i. Adopting the terminology of Section 2, morphemes correspond to target elements and characters to source elements. Each source element is aligned to exactly one target element and the alignment is monotonic. This means each source element can only be aligned to the same target element as the immediately preceding source element or alternatively it can be aligned to the next target element. Accordingly, we compute marginals, i.e. distributions over targets for each source element, by the forward-backward algorithm, same as Kim et al. (2017). The forward recursion is given by $\alpha_{1,1} = 1$ and

$$\alpha_{i,j} = \alpha_{i-1,j} \cdot (1 - p_{i-1}^{\text{seg}}) + \alpha_{i-1,j-1} \cdot p_{i-1}^{\text{seg}}$$

for $i > 1, j \geq 1$ where $p_i^{\text{seg}} \in (0; 1)$ is the predicted segmentation probability of the ith source element. Note that the first source element is always part of the first segment. Backward scores are computed as $\beta_{n,k} = 1$ and

$$\beta_{i,j} = \beta_{i+1,j} \cdot (1 - p_{i,j}^{\text{seg}}) + \beta_{i+1,j+1} \cdot p_{i,j}^{\text{seg}}$$

for $i < n$ and $j \leq k$. Finally, marginals are given by $\hat{\xi}_{i,j} = \frac{\alpha_{i,j} \cdot \beta_{i,j}}{\alpha_{n,k}}$. In practise, computations are performed in log-space.

Training For discretising segmentations, it is most convenient to simply choose the maximum likelihood segmentation, which corresponds to starting new segments at the $k - 1$ indices with maximum segmentation probabilities. The corresponding target segment representations are computed by $\mathbf{Y} = \xi \cdot \mathbf{X}$, where $\mathbf{X}$ is the matrix of source element representations. Note that we use the discrete assignments ξ for computing segment representations and Equation (1) for training.

In the case of interlinear glossing, distributions over labels for each morpheme are computed by a MLP taking morpheme representations, i.e. rows in $\mathbf{Y}$, as input. Loss, then, is the cross-entropy between predicted distributions over labels and ground-truth labels. Note that in this case, and in contrast to Section 3.1, we compute morpheme representations from contextualised character representations, and not from non-contextual embeddings, because we think that only sets of characters of shallow morpheme segments are not sufficient to compute the semantic information necessary for glossing, especially the correct translations.

This has two consequences, first of all the model is not transparent (i.e. interpretable), and character encodings may be "fuzzy" in the sense that information is locally spread across multiple characters which may obscure precise morpheme boundaries. A similar effect has been shown for sparse attention by Meister et al. (2021). We leave exploring more interpretable models similar to the model described in Section 3.1 and biasing models towards more precise morpheme segmentation to future work. In this work, our main focus is to provide shallow morpheme segmentation as additional predictions, not to build an entirely interpretable glossing model.

4 Evaluation

Here, we evaluate the methods presented in Section 3 by participating in the shared task on morphological inflection and in the shared task on interlinear glossing. Technical details of the experimental setup and hyperparameters are in Appendix A.

4.1 Baselines

For the morphological inflection shared task, the organisers provide a neural and a non-neural baseline. For the interlinear glossing shared task, the organisers provide a transformer-based neural baseline. Furthermore, we add a CTC-based sequence labelling model (Graves et al., 2006) as baseline. The CTC model encodes the source sentence on the character level by a BiLSTM encoder and predicts a label or blank from each character. Here, we exploit that the number of labels is the same as the number of morphemes, and each word has at least as many characters as morphemes.

4.2 Data Representation

For a detailed description of the shared task data, refer to the respective shared task overview papers. In the case of morphological inflection, we convert (lemma, tags, form) triples to (source, target) pairs by removing all punctuation from the tags and prepending the remaining sequence of tags to the lemma characters. Special pre- and postprocessing is applied to Japanese in order to eliminate Kanji, see Appendix C.

In the case of interlinear glossing, no modification is necessary for the closed track. However, for the open track, we replace the source text by hyphen-separated morphemes. We assume they contain more information than the original unsegmented text, and the unsegmented text does not add any information when the segmented morphemes are available. In this case, we also do not learn shallow segmentation, but predict a single label from each morpheme. The CTC baseline flexibly learns alignments of labels to characters in both cases. In both cases, we approach interlinear glossing as a sequence labelling problem.

4.3 Results

Morphological Inflection In Table 1, we report macro-averaged test set accuracy and edit distance. Full results for all languages achieved by our model and the baselines are in Appendix D. For clarity, we only report results of the best system for every participating team. Results show that our interpretable model loses on performance compared to more flexible neural models, such as the Transformer baseline. On 22 of 27 languages, the neural baseline beats our model. However, results also show that introducing interpretability does not have catastrophic consequences regarding performance. With some advantages in macro-averaged scores, our model performs roughly on par with the non-neural baseline, beating it on 14 of 27 languages. In summary, these results suggest that introducing interpretability to neural models causes some decrease in performance, but having neural interpretable models still gives better results than having interpretable non-neural models.

To illustrate patterns learned by our models, we show examples of the selected source symbols and condition symbols. The first example in Figure 1 is taken from the French (validation) dataset, namely the target inflection is

$$\textit{juger}\ \text{“to judge”}\ \xrightarrow{\text{V;COND;NOM(1,PL)}}\ \text{“jugerions”}.$$

	Accuracy ↑	ED ↓
Illinois	84.27	0.35
Baseline (Neural)	81.61	0.40
Ours	**76.91**	**0.58**
Arizona	72.45	0.75
Baseline (¬ Neural)	69.60	0.81

Table 1: Morphological Inflection: Best macro-averaged test set results for accuracy and edit distance (ED) of each team. Results of our model are highlighted in bold.

In this case, the prediction is correct. We can see how the model first selects characters of the stem to copy. Here, few if any other source symbols are selected as conditions. Then, for predicting the inflection suffix "-ions", the model selects tag symbols both to transduce and as conditions.

Next, we consider an example from the Italian dataset, namely

estraniarsi "to alienate onself"

$\xrightarrow{\text{V;IND;PRS;NOM(1,PL)}}$ "ci estranieremmo".

The corresponding selected symbols and conditions are shown in Figure 2. This example shows an interesting non-monotonic pattern, namely moving the reflexive pronoun "si" to the front and changing it to the correct number and person, in this case 1st plural. The model correctly captures this, as we can see from the selected transduction symbols (left side). Also, the model learned which conditions to select for changing the "s" in "si" to "c". After this transform, the model copies the stem by selecting stem characters as transduction symbols and conditions. Finally, the model generates the inflectional suffix by selecting mainly tags as transduction characters and conditions.

Interlinear Glossing In Table 2, we report macro-averaged word level and morpheme level test set accuracies. Both our additional CTC baseline and our morpheme-segmentation model, henceforth referred to as "morph", compare favourably to the transformer baseline. Furthermore, our morph model achieves better performance than competing models on track 1, where the unsupervised learning of morpheme segmentation is relevant, which shows that learning additional linguistically relevant structures can improve performance by injecting useful inductive biases in the model. Furthermore, we again conclude that using discrete structure as intermediate representations does not necessarily decrease performance catastrophically. Instead, it seems helpful in this case. Finally, we note that translations are not necessary for current glossing models to achieve strong performance, since we do not use them.

We also show examples of shallow segmentation learned by the morph model. We focus only on the segmentation, because this is the main contribution of our model. Note that all segmentations were learned in an unsupervised way for track 1 alongside the main objective, i.e. predicting the glossing line. For track 1, morphological segmentation is not given, unlike for track 2.

Because our model learns shallow segmentations, while ground-truth segmentations provided for track 2 are canonical segmentations, we can not conduct a quantitative analysis. Therefore, we restrict the analysis to anecdotal qualitative analysis of 2 example segmentions predicted by our models.

First, we consider a prediction for the following Natugu example:

(2)					
	Nedr	*rlilrdr*	*doa*	*nzpwxng*	.
	Ne-dr	r-li-lr-dr	doa	nz-pwx-ng	.
	ne-dr	r-li-r-dr	doa	nz-pwx-ngq	.

"The two of them had four children."

Morphemes are separated by hyphens "-". The predicted segmentation is in the second line, and the ground-truth segmentation is in the third line. The predicted segmentation differs from the ground-truth, because it copies characters and therefore can not change capitalisation, and two morphemes ("lr" → "r" and "ng" → "ngq") are normalised in the ground-truth segmentation, so that they differ from their surface form.

Next, we consider a Uspanteko example:

(3)						
	i	*tiyuq*	*sol*	*ji'*	*tren*	*tib'ek*
	i	t-iyuq	sol	ji'	t-r-en	t-ib'e-k
	i	ti-yu'	sol	ji'	t-r-en	ti-b'e-k

"Y llega solo asi hace se va."

Again, the predicted segmentation is in the second line. In two cases the vowel "i" is assigned to the wrong morpheme, but the predicted glossing line (not shown) is still correct. In this case, incorrect segmentation is apparently not a problem for the subsequent classification of morphemes, but in other cases this may cause problems. Here, we

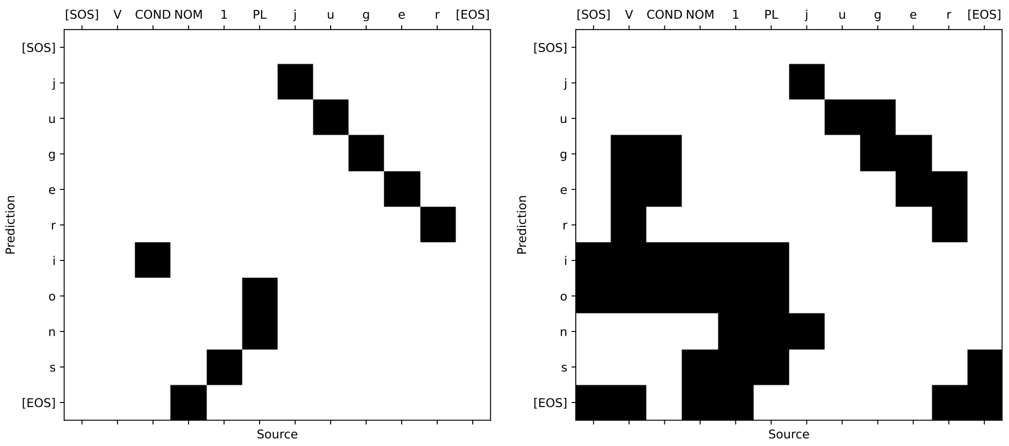

Figure 1: Example for French *juger* “to judge”. The target prediction is “juger” $\xrightarrow{\text{V;COND;NOM(1,PL)}}$ “jugerions”. The prediction is correct in this case. On the left side, we show the single selected symbols for transduction, on the right side we show the additionally selected condition symbols. Because we use hard attention, attention scores can only be 0 or 1, and we can present them in a black-and-white style.

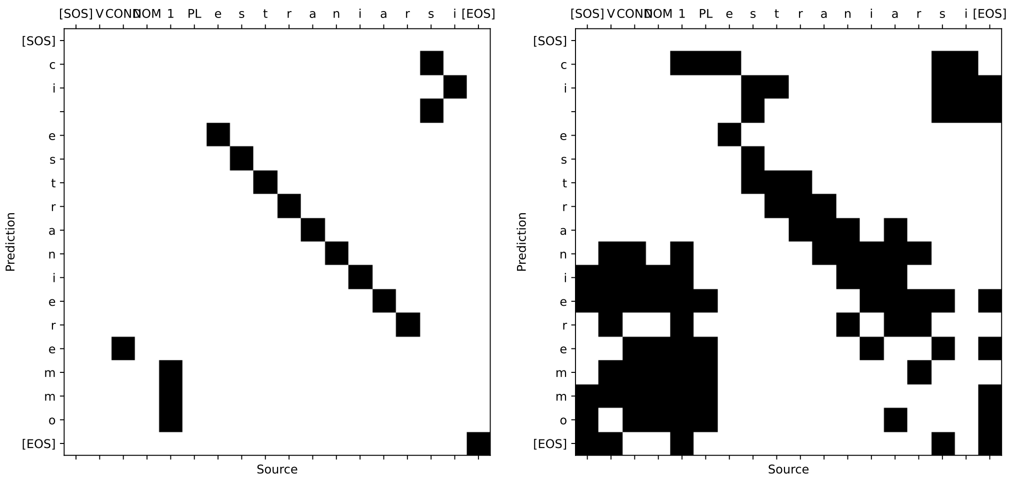

Figure 2: Example for Italian *estraniarsi* “to alienate onself”. The target prediction is “estraniarsi” $\xrightarrow{\text{V;COND;NOM(1,PL)}}$ “ci estranieremmo”. The prediction is correct in this case. On the left side, we show the single selected symbols for transduction, on the right side we show the additionally selected condition symbols.

	Track 1		Track 2	
	Word Level	Morph. Level	Word Level	Morph. Level
Ours (Morph)	71.30	62.55	76.56	84.21
Ours (CTC)	68.01	60.24	74.43	78.03
Baseline	47.31	33.60	59.14	67.69

Table 2: Interlinear Glossing: Macro-averaged test set accuracy for our models and the baseline. Higher is better.

can see that contextualised character encodings can bypass the discrete morpheme segmentation step.

5 Related Work

Much recent work in character-level transduction aims at making models both more interpretable and stronger by using sparse (Peters and Martins, 2019, 2020) or hard attention (Aharoni and Goldberg, 2017; Wu et al., 2018; Wu and Cotterell, 2019; Makarov and Clematide, 2018b,a). Modifying the attention mechanism is necessary, because for soft attention, which does not realise hard alignment decisions, the relation of model outputs and attention weights is not fully understood (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019). Hard attention does not suffer from this problem, because it realises hard alignment decisions, so that we exactly know which information influences the output of attention. Especially if the main purpose of models is to gain insights into the data and not mainly to achieve better performance on modelling it, interpretability of models is crucial.

However, working with hard attention is notoriously difficult, because it introduces non-differentiability into models. This means that the sophisticated machinery developed for gradient-based optimisation of deep neural models fails in this case. The most popular but also most expensive approach to train hard attention mechanisms is marginalising all alignments (Yu et al., 2016; Raffel et al., 2017; Wu et al., 2018; Wu and Cotterell, 2019) or approximating the marginalisation (Shankar et al., 2018). Different approaches to approximate gradients instead of having exact gradients by marginalisation are using reinforcement learning (Xu et al., 2015; Makarov and Clematide, 2018a), reparametrising discrete distributions or working with continuous relaxations (Jang et al., 2017; Maddison et al., 2017), and perturbation based gradient approximators (Niepert et al., 2021). While all methods to use hard attention with gradient-based optimisation come with problems, we find straight-through gradient estimators (Bengio et al., 2013) very effective to compute informative discrete intermediate representations even for complicated attention scenarios. Similar results have been reported for other fields as well, both practically (Sahoo et al., 2022) and in theoretical analysis (Yin et al., 2019). Therefore, we propose two models, one for interpretable sequence transduction and one for joint segmentation and segment classification, based on straight-through gradient estimators.

6 Conclusion

In this work, we propose to optimise hard attention as building block in deep neural networks, which in principle is non-differentiable, by straight-through gradient estimation. In particular, we describe applications to interpretable sequence-to-sequence models and sequence segmentation models. We evaluate our approaches on two important tasks in computational morphology, namely morphological inflection and interlinear glossing which are shared tasks of the ACL SIGMORPHON 2023 workshop. Our approaches achieve good results in morphological inflection despite of constrained expressivity compared to fully differentiable models, and strong results in interlinear glossing. These results provide encouraging evidence that learning interpretable intermediate representations by deep neural network does not necessarily lead to intolerable sacrifices in performance. We hope that future work can benefit from these insights by combining interpretable representations and the generalisation abilities of neural models for scientific discovery.

Acknowledgements

We thank the organisers of both shared tasks for their efforts and for their help during training and evaluation phases. Also, we thank Çağrı Çöltekin for his feedback on a draft version of this paper and Leonard Salewski for a helpful discussion in the early phase of this project.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Gianni Brauwers and Flavius Frasincar. 2023. A general survey on attention mechanisms in deep learning. *IEEE Trans. Knowl. Data Eng.*, 35(4):3279–3298.

Nadia Burkart and Marco F. Huber. 2021. A survey on the explainability of supervised machine learning. *J. Artif. Intell. Res.*, 70:245–317.

Chundra Cathcart and Florian Wandl. 2020. In search of isoglosses: continuous and discrete language embeddings in Slavic historical phonology. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 233–244, Online. Association for Computational Linguistics.

Bernard Comrie, Martin Haspelmath, and Balthasar Bickel. 2008. The leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses. *Department of Linguistics of the Max Planck Institute for Evolutionary Anthropology & the Department of Linguistics of the University of Leipzig*.

Omer Goldman, Khuyagbaatar Batsuren, Khalifa Salam, Aryaman Arora, Garrett Nicolai, Reyt Tsarfaty, and Ekaterina Vylomova. 2023. SIGMORPHON–UniMorph 2023 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 20th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Toronto, Canada. Association for Computational Linguistics.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM.

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 Comput.*, 9(8):1735–1780.

Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. 2020. Explainable AI methods - A brief overview. In *xxAI - Beyond Explainable AI - International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, volume 13200 of *Lecture Notes in Computer Science*, pages 13–38. Springer.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. 2020. Fantastic generalization measures and where to find them. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 961–967, Austin, Texas. Association for Computational Linguistics.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jingwen Li and Leander Girrbach. 2022. Word segmentation and morphological parsing for sanskrit. *arXiv preprint arXiv:2201.12833*.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Hairong Liu, Zhenyao Zhu, Xiangang Li, and Sanjeev Satheesh. 2017. Gram-ctc: Automatic unit selection and target decomposition for sequence labelling. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2188–2197. PMLR.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Peter Makarov and Simon Clematide. 2018a. Imitation learning for neural morphological string transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2877–2882, Brussels, Belgium. Association for Computational Linguistics.

Peter Makarov and Simon Clematide. 2018b. Neural transition-based string transduction for limited-resource setting in morphology. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 83–93, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Clara Meister, Stefan Lazov, Isabelle Augenstein, and Ryan Cotterell. 2021. Is sparse attention more interpretable? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 122–129, Online. Association for Computational Linguistics.

Mathias Niepert, Pasquale Minervini, and Luca Franceschi. 2021. Implicit MLE: backpropagating through discrete exponential family distributions. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 14567–14579.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Ben Peters and André F. T. Martins. 2019. IT–IST at the SIGMORPHON 2019 shared task: Sparse two-headed models for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 50–56, Florence, Italy. Association for Computational Linguistics.

Ben Peters and André F. T. Martins. 2020. One-size-fits-all multilingual models. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 63–69, Online. Association for Computational Linguistics.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2837–2846. PMLR.

Sukrut Rao, Moritz Böhle, and Bernt Schiele. 2022. Towards better understanding attribution methods. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10213–10222. IEEE.

Subham Sekhar Sahoo, Anselm Paulus, Marin Vlastelica, Vít Musil, Volodymyr Kuleshov, and Georg Martius. 2022. Backpropagation through combinatorial algorithms: Identity with projection works. *arXiv preprint arXiv:2205.15213*.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 640–645, Brussels, Belgium. Association for Computational Linguistics.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Confer-*

ence on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1530–1537, Florence, Italy. Association for Computational Linguistics.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438, Brussels, Belgium. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2048–2057. JMLR.org.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley J. Osher, Yingyong Qi, and Jack Xin. 2019. Understanding straight-through estimator in training activation quantized neural nets. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316, Austin, Texas. Association for Computational Linguistics.

A Experimental Setup

Here, we describe our experimental setup, including hyperparameters, in detail. Also note that our code is available on GitHub. Code for morphological inflection is here: `https://github.com/LGirrbach/sigmorphon-2023-glossing`. Code for interlinear glossing is here: `https://github.com/LGirrbach/sigmorphon-2023-inflection`. All models are implemented in `PyTorch` (Paszke et al., 2019) and `pytorch_lightning`.[3]

In all cases, models are optimised using the AdamW optimizer (Loshchilov and Hutter, 2019) without weight decay, learning rate is 0.001, and all other parameters are the PyTorch defaults. We use an exponential decay learning rate scheduler which multiplies the learning rate by factor γ after each epoch. We tune γ for each combination of model and language. Furthermore, we clip gradients with absolute value greater than 1.

Models are trained exclusively on the training splits provided by the shared task organisers. After each training epoch, generalisation performance is estimated by performance on the validation split. In the case of Morphological Inflection, our main metric is normalised edit distance (lower is better). In the case of Interlinear Glossing, our main metric is accuracy of correctly predicted glossing lines (higher is better). If performance does not improve for 3 epochs, training is stopped. Only the best model checkpoint, i.e. the checkpoint 3 epochs before ending training, is retained.

A.1 Hyperparameter Tuning

For each combination of language and model, we optimise hyperparameters independently. The tuned hyperparameters and their corresponding value ranges are in Table 3. Note that other than stated there, the minimum batch size for Morphological Inflection models is always 4. Also, minimum batch size for Arapaho and Uspanteko languages (Interlinear Glossing) is 16. The maximum batch size for Arapaho and Uspanteko is 128. The maximum batch size for Gitksan (Interlinear Glossing) is 16.

In each case, we sample 50 sets of hyperparameters using the `optuna` library (Akiba et al., 2019). For each sampled set, a model is trained and the performance on the validation set is recorded. After sampling 50 sets, the set that resulted in the best performance on the validation set is saved. For training models for submission of results to the shared tasks and all other analyses, we exclusively use hyperparameter that were found best in this hyperparameter study. Since we tune parameters for many models, we do not report the results here. However, they are available in our GitHub repositories.

A.2 Main Evaluation

For submitting results to the shared tasks and further analyses, we retrain 5 models for each combination of model type and language. The models use the hyperparameters from the tuning study described in Appendix A.1. However, they have different initialisations and may therefore perform differently. For submitting results, we select the

[3] `https://github.com/Lightning-AI/lightning/`

Parameter Name	Range
# LSTM Layers	$\{1, 2\}$
Hidden Size	$[64; 512]$
Dropout	$[0.0; 0.5]$
Scheduler γ	$[0.9; 1.0]$
Batch Size	$[2; 64]$

Table 3: Ranges for values of tuned hyperparameters. Note that batch size ranges differ in some cases.

model the model with best performance on the validation set. Having multiple copies of the same model type trained with the same hyperparameters is useful, because especially in low-resource scenarios different initialisations can have a relevant impact on the generalisation performance. On the one hand, it is necessary to estimate this variance to see how robust models are, on the other hand this helps mitigating spurious effects in the analyses.

B Sequence Transduction Model: Detailed Description

Here, we provide a more detailed description of the sequence transduction model (see Section 3.1).

For training, we have two paired sequences $s = s_1, \ldots, s_n$ and $t = t_1, \ldots, t_m$, representing the source and target sequence, respectively. The goal is to maximise the likelihood of transducing the source sequence s to the target sequence t.

The first step is to represent all symbols in both sequences by high-dimensional non-contextual vectors, i.e. embeddings. Thus, we arrive at embedded sequences $\mathbf{s} = \mathbf{s}_1, \ldots, \mathbf{s}_n$ and $\mathbf{t} = \mathbf{t}_1, \ldots, \mathbf{t}_m$, with $\mathbf{s}_i, \mathbf{t} \in \mathbb{R}^d$. Here, boldfaced lowercase variables represent vectors. Furthermore, we denote the $n \times d$ matrix with source symbol embeddings as rows as $\mathbf{S} \in \mathbb{R}^{n \times d}$ and we denote the $m \times d$ matrix with all target symbol embeddings as $\mathbf{T} \in \mathbb{R}^{m \times d}$. Note, that embeddings of source symbols and target symbols are optimised independently of each other, i.e. they are not paired.

In the next step, we encode the source sequence by a *bidirectional* LSTM and arrive at a contextual representation $\mathbf{h}_i^s$ for each source symbol with index i, $1 \leq i \leq n$. Likewise, we encode the target sequence using a *unidirectional* LSTM and denote the autoregressive encoding of the symbol with index j, $1 \leq j, \leq m$ as $\mathbf{h}_j^t$. Formally, we can write

$$\mathbf{h}_i^s = \mathrm{BiLSTM}(\mathbf{s}_i \mid \mathbf{s}_1, \ldots, \mathbf{s}_n) \tag{3}$$

$$\mathbf{h}_j^t = \mathrm{LSTM}(\mathbf{t}_j \mid \mathbf{t}_1, \ldots, \mathbf{t}_{j-1}) \tag{4}$$

and representing all contextualised representations as matrices:

$$\mathbf{H}^s = \mathrm{BiLSTM}(\mathbf{S}) \tag{5}$$

$$\mathbf{H}^t = \mathrm{LSTM}(\mathbf{T}) \tag{6}$$

So far, this formulation is the same as conventional LSTM-based encoder-decoder models. However, the attention mechanism is different. According to the descriptions in Section 2 and Section 3.1, we define attention matrices as follows:

$$\mathbf{Z} = \mathbf{H}^s \cdot (\mathbf{H}^t)^T \tag{7}$$

$$\hat{\xi}_{\text{symbol}} = \mathrm{softmax}(\mathbf{Z}) \tag{8}$$

$$\hat{\xi}_{\text{cond}} = \sigma(\mathbf{Z}) \tag{9}$$

where $\mathbf{Z} \in \mathbb{R}^{n \times m}$ represents the unnormalised dot-product attention scores, $\mathrm{softmax}$ is applied to columns of $\mathbf{Z}$ and the sigmoid function σ is applied elementwise. In fact, it seems counterintuitive to use the same unnormalised scores in both attention heads, but this works well in practise. From these attention matrices, we obtain the discretised alignment matrices ξ_{symbol} and ξ_{cond} by sampling columnwise one-hot vectors in the case of ξ_{symbol} and elementwise values $\in \{0, 1\}$ for ξ_{cond}. Then, we calculate context vectors as

$$\mathbf{Y}_{\text{symbol}} = \xi_{\text{symbol}}^T \cdot \mathbf{S} \tag{10}$$

$$\mathbf{Y}_{\text{cond}} = \xi_{\text{cond}}^T \cdot \mathbf{S} \tag{11}$$

Note, that here we use the discretised alignment matrices ξ_{symbol} and ξ_{cond} in place of the real-valued matrices $\hat{\xi}_{\text{symbol}}$ and $\hat{\xi}_{\text{symbol}}$. Because discretisation is non-differentiable, we instead use a way of calculation that enables straight-through gradient estimation, namely Equation (1):

$$\begin{aligned}\mathbf{Y}_{\text{symbol}} = {} & \hat{\xi}_{\text{symbol}}^T \cdot \mathbf{S} \\ & - \mathrm{sg}\left((\hat{\xi}_{\text{symbol}} - \xi_{\text{symbol}})^T \cdot \mathbf{S}\right)\end{aligned} \tag{12}$$

$$\begin{aligned}\mathbf{Y}_{\text{cond}} = {} & \hat{\xi}_{\text{cond}}^T \cdot \mathbf{S} \\ & - \mathrm{sg}\left((\hat{\xi}_{\text{cond}} - \xi_{\text{cond}})^T \cdot \mathbf{S}\right)\end{aligned} \tag{13}$$

Finally, we predict a distribution over target symbols by a MLP from the concatenation of $\mathbf{Y}_{\text{symbol}}$ and $\mathbf{Y}_{\text{cond}}$ according to Equation (2):

$$p_j(\bullet \mid s_1, \ldots, s_n) = \text{MLP}([\mathbf{Y}_j^{\text{symbol}}, \mathbf{Y}_j^{\text{cond}}]) \tag{14}$$

where $p_j(\bullet \mid s_1, \ldots, s_n)$ indicates the distribution over target symbols at prediction position with index j. Here, the concrete ground truth target symbol is t_j.

To train the model, we use the typical supervised objective, namely minimising the cross-entropy between predicted categorical distributions over target symbols and the one-hot encoded ground truth target symbol at each prediction position. This also means we use teacher forcing for sequential predictions. During inference, t is predicted in a step-wise fashion by greedy decoding.

However, note that in addition to the setup described above, we also use the typical attention mechanism in a multi-tasking fashion, but only during training. We do not use contextualised source symbol representations for inference. The reason why we still use the typical loss as auxiliary loss is that the gradient signal on $\mathbf{H}^s$ and $\mathbf{H}^t$ is weak when their only role is to calculate $\mathbf{Z}$. In this case, we calculate

$$\mathbf{C} = \left(\text{softmax}\left(\mathbf{H}^s \cdot (\mathbf{H}^t)^T\right)\right)^T \cdot \mathbf{H}^s \tag{15}$$

where softmax is applied to columns, and then predict distributions over target symbols as

$$p_j^{\text{aux}}(\bullet \mid s_1, \ldots, s_n) = \text{MLP}(\mathbf{C}) \tag{16}$$

so that we can also calculate the cross-entropy loss on $p_j^{\text{aux}}(\bullet \mid s_1, \ldots, s_n)$ and differentiate model parameter w.r.t. the sum of both losses during training. Note, again, that this is only to stabilise training by optimising the contextual representations, and does not affect the model architecture for inference.

C Preprocessing of Japanese

The Japanese writing system includes 4 alphabets, namely Latin characters (Romaji), Chinese characters (Kanji), and two syllabic scripts (Katakana and Hiragana) that were derived from Chinese characters by simplification and standardisation. The Japanese dataset provided with the shared task contains Kanji, Hiragana, and Katakana. Kanji constitute a problem for character-level models, because they are effectively an open set. The number of Kanji taught in Japanese schools is already ≈ 2000, to which variants, obsolete Kanji, and special Kanji only used in names may be added. Therefore, a model for Japanese language data should have the possibility to process previously unseen Kanji.

In the case of morphological inflection, however, this problem may be ignored, because Kanji are never altered in inflection. Instead, inflectional suffixes are expressed in Hiragana. Kanji may still appear in stems, and have therefore be dealt with.

We apply the following preprocessing: We replace all Kanji by a special placeholder symbol K that is the same for every Kanji. This applies to both source lemmas and target forms. In the case of successful prediction of target lemmas, the number of Kanji in the target form is the same as the number of Kanji in the source lemma. Therefore, we copy Kanji from the source by replacing the predicted placeholders. The order of Kanji in source and target does not change. In case of predicting fewer placeholders in the predicted form than there are Kanji in the source lemma, which is an error, we copy as many Kanji as there are placeholders in the predicted form from left to right. In case of predicting more placeholders in the predicted form than there are Kanji in the source lemma, which also is an error, we leave additional placeholders unchanged, i.e. we do not change the predicted placeholder symbol, but still copy as many Kanji as possible from left to right.

D Full Results

Morphological Inflection In Table 5, we report the official test set accuracies achieved by our model for all languages. For comparison, we also show results of the neural and the non-neural baseline. Likewise, we report official test set edit distances in Table 6.

These results show some interesting patterns. For example, the neural baseline performs worst on Japanese, which we assume is an effect of the alphabet (see Appendix C). Therefore, our proposed pre- and postprocessing for Japanese is important. Alternatively, a copy mechanism could be used.

Another trend is that our model performs strongly on semitic languages (e.g. afb, amh, arz, heb, heb_unvoc), especially compared to the non-neural baseline. Here, non-concatenative morphology gives our model an advantage, because usually individual transforms do not involve multiple characters, such as suffix ngrams. Remember that our model can only predict exactly one form character

	Track 1				Track 2			
	Word Level		Morph. Level		Word Level		Morph. Level	
	CTC	Morph	CTC	Morph	CTC	Morph	CTC	Morph
Arapaho	77.90	78.79	76.56	78.57	85.12	85.80	90.93	91.37
Tsez	80.96	80.94	70.29	73.95	85.68	85.79	91.16	92.01
Gitksan	04.69	21.09	09.26	11.72	13.80	26.56	17.08	50.22
Lezgi	78.10	78.78	62.03	62.10	85.44	83.41	83.45	87.61
Natugu	80.20	81.04	56.38	56.32	87.83	87.92	90.17	92.32
Nyangbo	85.34	85.05	86.74	85.24	85.90	87.98	89.96	91.40
Uspanteko	68.86	71.01	60.42	62.55	77.21	78.46	83.45	84.51

Table 4: Interlinear glossing: Official test set accuracies for all languages. Higher is better. CTC refers to our CTC-based baseline model, "Morph" refers to our model that learns morphological segmentation in a unsupervised way for track 1. In track 2, we simply use the provided representation.

from each combination of transduction lemma character and condition set. This shows that our model is able to capture complex patterns, but may not be optimal to generate extensive surface transforms.

Interlinear Glossing In Table 4, we report official test set accuracies achieved by our models for all languages. In most cases, our Morph model is superior to the CTC model, which confirms the benefit of morpheme segmentations for the task of interlinear glossing. The same conclusion is supported by the stark differences between track 1 and track 2. In track 2, performance is generally much better, especially when looking at morpheme level accuracies. Remember that we only use the ground-truth morpheme segmentation as additional information in track 2. We conclude that research in morpheme segmentation will be useful for computational models for interlinear glossing as well.

	Ours	Neural	¬ Neural
afb	75.8	80.1	30.8
amh	83.8	82.2	65.4
arz	87.6	89.6	77.9
bel	56.3	74.5	68.1
dan	85.7	88.8	89.5
deu	74.5	83.7	79.8
eng	96.0	95.1	96.6
fin	67.6	85.4	80.8
fra	67.9	73.3	77.7
grc	36.7	54.0	52.6
heb	82.7	92.0	30.9
heb(2)	81.3	83.2	64.5
hun	75.9	80.5	74.7
hye	85.9	91.0	86.3
ita	84.7	94.1	75.0
jap	95.3	26.3	64.1
kat	70.5	84.5	82.0
klr	96.4	99.5	54.5
mkd	86.7	93.8	91.6
nav	53.6	52.1	35.8
rus	82.1	90.5	86.0
san	54.5	66.3	62.2
sme	58.5	74.8	56.0
spa	88.7	93.6	87.8
sqi	71.5	85.9	83.4
swa	94.7	93.7	60.5
tur	81.8	95.0	64.6

Table 5: Official test set accuracies for all languages. Higher is better. "Neural" and "¬ Neural" refer to the neural and non-neural baseline, respectively. "heb(2)" refers to the `heb_unvoc` dataset.

	Ours	Neural	¬ Neural
afb	0.49	0.38	1.47
amh	0.22	0.24	0.59
arz	0.24	0.22	0.46
bel	1.31	0.64	0.90
dan	0.34	0.25	0.17
deu	1.00	0.48	0.80
eng	0.07	0.09	0.06
fin	0.63	0.21	0.26
fra	0.86	0.45	0.37
grc	1.43	1.00	1.04
heb	0.44	0.21	1.82
heb(2)	0.30	0.28	0.48
hun	0.57	0.44	0.47
hye	0.43	0.20	0.30
ita	0.43	0.12	0.75
jap	0.09	1.20	0.80
kat	0.79	0.35	0.51
klr	0.05	0.00	0.84
mkd	0.38	0.09	0.15
nav	1.37	1.55	1.88
rus	0.56	0.38	0.46
san	1.01	0.71	0.90
sme	0.88	0.65	0.89
spa	0.31	0.10	0.17
sqi	0.74	0.27	0.38
swa	0.06	0.06	4.10
tur	0.58	0.17	0.87

Table 6: Official test set edit distances for all languages. Lower is better. "Neural" and "¬ Neural" refer to the neural and non-neural baseline, respectively. "heb(2)" refers to the `heb_unvoc` dataset.