Interpretability for morphological inflection: from character-level predictions to subword-level rules

Tatyana Ruzsics **Olga Sozinova** **Ximena Gutierrez-Vasques** **Tanja Samardžić**
Text Group, URPP Language and Space,
University of Zurich, Switzerland
{tatiana.ruzsics, olga.sozinova, ximena.gutierrezvasques, tanja.samardzic}@uzh.ch

Abstract

Neural models for morphological inflection have recently attained very high results. However, their interpretation remains challenging. Towards this goal, we propose a simple linguistically-motivated variant to the encoder-decoder model with attention. In our model, character-level cross-attention mechanism is complemented with a self-attention module over substrings of the input. We design a novel approach for pattern extraction from attention weights to interpret what the model learn. We apply our methodology to analyze the model's decisions on three typologically-different languages and find that a) our pattern extraction method applied to cross-attention weights uncovers variation in form of inflection morphemes, b) pattern extraction from self-attention shows triggers for such variation, c) both types of patterns are closely aligned with grammar inflection classes and class assignment criteria, for all three languages. Additionally, we find that the proposed encoder attention component leads to consistent performance improvements over a strong baseline.

1 Introduction

With the rise of deep learning, neural networks have been nowadays used in the process of decision making in various domains, as different as trading, medicine and government. Ethical considerations of such decisions have led to an increasing need for interpreting neural models which is a vivid research topic in machine learning community (Lipton, 2018; Gilpin et al., 2018). Although interpretability research in NLP is partly driven by ethics (Jacovi and Goldberg, 2020), there is a growing body of work exploring what linguistic properties emerge in neural models (Belinkov and Glass, 2019; Manning et al., 2020). The latter line of work aims to aid and scale up linguistic research, which is also the topic of this paper. Linguistics research focuses on uncovering patterns and regularities in language. Retrieving and analyzing structures of languages learned by neural agents can systematize our knowledge and, ideally, help us to come up with new regularities. Recent advances (Schrimpf et al., 2020) in testing hypotheses about human language processing using the growing suite of modern interpretable NLP are, indeed, inspiring, but still relatively limited to few languages. To scale up linguistic research, we require truly language-independent models developed for languages other than English (Bender, 2011). In return, understanding the model's decisions can lead to new ideas, how to improve the performance of the model on hard cases, i.e. on a particular linguistic phenomenon or language.

In our work, we concentrate on interpretability methods suitable for examining what knowledge of inflection morphology is captured by neural networks. Specifically, we consider a neural model that learns a mapping from a lemma and an abstract morpho-syntactic definition (MSD) to its inflected form. MSD comprises a part-of-speech (POS) tag as well as language-specific inflection tags. For example, given the Italian lemma *scolorire* "discolor" and MSD *V;IND;PRS;3;PL* (verb, 3rd person plural present indefinite form), the output is the word form *scoloriscono*. Datasets for this task of morphological reinflection are available for many languages and provide an opportunity to study a broad set of inflection phenomena.

Character-level encoder-decoder neural models with attention achieved very high performance on this task across many languages (see Cotterell et al., 2017, 2018; Vylomova et al., 2020 for the results of the recent shared tasks). Nevertheless, this class of models is typically not interpreted, and if it is so, the interpretation is limited to visualizing attention heatmaps on selected examples (see e.g. Aharoni and Goldberg 2017; Peters and Martins 2019).

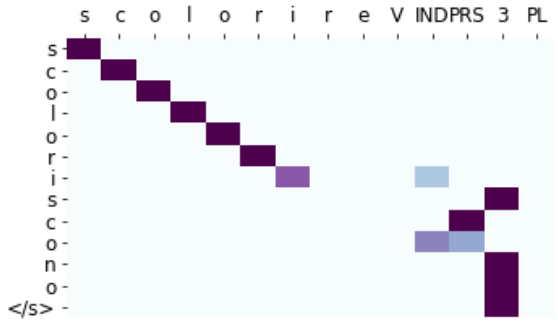

Figure 1: Example of a heatmap visualizing attention weights learned by the model of Peters and Martins (2019). The inflected form is generated from top to bottom.

We argue that per-example heatmaps provide very limited insight into what neural agent learns about a specific inflection phenomenon and how neural learning process can be related, in a systematic way, to the linguistic theory. Indeed, consider the previous Italian example: how would humans reason to convert the lemma to its inflected form?[1] In this work, we assume that humans apply the rules of grammar (implicit for native speakers, and explicit for language learners) when they perform this task. Specifically to our example, human reasoning could look as follows: the verb *scolorire* ends with the suffix *-ire* which determines its inflection class[2], according to which we construct the inflected form by copying the stem *scolor-* and adding an inflection suffix *-iscono*. What does a typical character-level model of an encoder-decoder class do? We visualize learned attention weights for an example of such a model in Fig. 1. By analyzing the most prominent alignments, we conclude that the model's character-level decisions can be combined into a) copying a substring of characters corresponding to the stem and b) generating characters for a substring corresponding to the inflection suffix. However, the decision why the model chose a particular inflection class is not visible.

To reach interpretability of models for inflection, we require methods that satisfy three conditions. First, the inflection model's decisions have to be aligned more closely to human reasoning by separating two kinds of operations: determining inflection class versus generating a string (given the assignment to a class). Second, a systematic analysis of model's decisions requires the extraction of inflection rules that are interpretable to humans. Finally, both of these factors require working with subword units rather than individual characters, the latter being the prevailing practice for inflection models.

In this paper, we propose a methodology - subword modification for a typical inflection model and interpretation method - which meet all three requirements. Our experiments on three typologically-different inflection phenomenon show that the linguistic rules elicited with our framework are highly consistent with linguistic knowledge (approximated by grammars). We evaluate the effectiveness of the proposed subword modification and find that, apart from its direct impact on interpretability, it leads to consistent performance improvements. To facilitate the use of our methods for linguistic research, we share our code[3].

2 Methodology: Interpretability for Inflection

How to make a character-level neural model for inflection more interpretable? We take the stance that, to make current models more interpretable, we should analyze their decisions in terms of subwords, i.e. clusters of characters rather than individual characters. Interpreting character decisions is outside of human intuition because of the double articulation principle (Martinet, 1967) which postulates that single phones are uninterpretable to humans, whereas the clusters of them form a mental linguistic representation of meaning in the speaker's mind. In writing, this distinction maps to the one between single characters and morphemes.[4]

From this perspective, we formulate the following desired properties for retrieving linguistic structures of inflection. For a given grammatical category (e.g. MSD *V;IND;PRS;3;PL* from the previous example in Italian),

[1] This is something that speakers implicitly (maybe, we do not know this for a fact) perform every time they use a word. This is also often an explicit task that language learners perform in the process of acquiring a new language.

[2] There are three inflection classes in Italian (Table 1). Verbs ending on *-ire* select between two classes: one (in the example) is more frequent in terms of types of verbs that belong to it, whereas the other one (*-ere* class) is selected by some very frequent verbs ending on *-ire*.

[3] https://github.com/tatyana-ruzsics/interpretable-inflection

[4] In this work, we refer to morphemes in their most general sense - apart from functional words and affixes we consider morphological processes, e.g. infixation and reduplication

	ARE	ERE/IRE	IRE
1SG	O	O	ISCO
2SG	I	I	ISCI
3SG	A	E	ISCE
1PL	IAMO	IAMO	IAMO
2PL	ATE	ETE	ITE
3PL	ANO	ONO	ISCONO

Table 1: Italian verbal inflectional classes, present tense

P1: identify substrings in an inflected form correspondig to morpheme(s) attributed to this category // *scolor**iscono***

P2: identify variation (dataset-wide) in the form of the morpheme (morphological class), e.g. all other substrings attributed to the same category // *-ano, -ono, -iscono*

P3: indicate whether triggers for such variation can be attributed to particular substrings of the lemma // *scolor**ire***

P4: identify triggers (dataset-wide) for each inflection class identified in P2 // *-are, -ere, -ire*

We propose to extract human-interpretable rules from an encoder-decoder model with attention. Specifically, to make it more interpretable, we modify such model by complementing the cross-attention mechanism with a novel component (§3) for self-attention over the subwords of the lemma. The task of this component is to help identify the morphological class. To extract the rules, we design a pattern extraction method (§4) that aggregates learned attention weights a) over a span of characters in a word and b) over a range of words in the same inflection category. Pattern extraction applied to character-level cross-attention weights retrieves linguistic rules satisfying requirements P1 and P2, whereas pattern extraction applied to subword-level self-attention weights, targets the requirements P3 and P4.

2.1 Case studies

To demonstrate the use of our approach for linguistic research, we will analyze how well patterns extracted with our proposed methodology align with human knowledge of typologically different phenomena. In morphological typology, cross-linguistic strategies to define the form and meaning of morphemes are described by typological parameters (Shopen, 1985; Dryer and Haspelmath, 2013; Bickel and Nichols, 2007) that separate different dimensions of the strategies. To select typologically different languages for our study, we focus on fusion and flexivity dimensions. Fusion classifies how easy it is to find a boundary between a morpheme and its phonological host and can take the following values: isolating (separate phonological word), concatenative (segmentable dependent morphemes) and nonlinear (not segmentable morphemes). Flexivity indicates whether variation in morpheme form can be explained by phonological processes (nonflexive) or not (flexive).

In our case studies, we consider verb conjugation rules and select three languages covering different degrees of fusion and flexivity: Finnish, Italian and Tagalog. In Finnish and Italian, morphemes are separable (concatenative fusion), whereas inflection in Tagalog is formed with affixes, including infixes, and reduplication (nonlinear fusion). Morphemes in Finnish can change their shape because of vowel harmony (nonflexive case). Forms of morphemes in Italian and Tagalog are selected by lexical context (flexive case) but distinctly. Italian verbs are conjugated with respect to three inflection classes defined by the lemma's ending (*-are*, *-ire*, and *-ere*, see Table 1), whereas assignment to an inflection class in Tagalog has no explicit rule.

3 Neural Model Cross-Att^{ch}Self-Att^{sub}

In this section, we introduce our novel component for **self-attention over subwords** of the lemma (**Self-Att^{sub}**). This module can be integrated into any variant of encoder-decoder system for inflection with **character-level cross-attention** (**Cross-Att^{ch}**). In this work, we show such integration to a sparse two-headed model of Peters and Martins (2019).[5] To explain the two-headed attention mechanism of the baseline model as well as our novel attention component, we first introduce the terminology of an abstract attention head module.

Attention Head Given an input sequence of vectors $\mathbf{H} = \mathbf{h}_1 \dots \mathbf{h}_J$, $\mathbf{h}_k \in \mathbb{R}^{D1}$ and a query vector $\mathbf{q} \in \mathbb{R}^{D2}$, attention head module computes two components: attention weights $a \in \mathbb{R}^J$ and an attention head vector $\mathbf{c} \in \mathbb{R}^{D1}$:

$$\alpha_j = \mathbf{q}^\intercal \mathbf{W}_a \mathbf{h}_j, \quad \mathbf{c} = \sum a_j \mathbf{h}_j \qquad (1)$$

where attention weights a are obtained by a mapping function from real values to probabilities, applied to alignment scores α. Following Peters and

[5]This model was among the winners of SIGMORPHON 2019 shared task (ranked first in terms of edit distance)

Martins (2019), we use sparsemax activations[6] as a mapping function. Hereafter we refer to the construction of an attention head vector $\mathbf{c}$ as scoring a sequence of vectors $\mathbf{H}$ with a query vector $\mathbf{q}$.

Baseline Cross-Att^{ch} (Peters and Martins, 2019) Input lemma and MSD sequences are represented by separate bi-LSTM encoder states: $\mathbf{H}^u$ encodes characters in lemma, and $\mathbf{H}^v$ encodes tags in MSD. The decoder is a unidirectional LSTM with input feeding (Luong et al., 2015). At each prediction time t, it computes a hidden state $\mathbf{s}_t$ which is followed by the construction of two attention heads $\mathbf{u}_t$ and $\mathbf{v}_t$: one for lemma and one for MSD. They are calculated by scoring the respective representations, $\mathbf{H}^u$ and $\mathbf{H}^v$, with a query - decoder state s_t. The two attention heads are used to compute separate candidate attentional decoder states:

$$\tilde{\mathbf{s}}_{tu} = \texttt{tanh}(\mathbf{W}_u[\mathbf{u}_t; \mathbf{s}_t]) \quad (2)$$

$$\tilde{\mathbf{s}}_{tv} = \texttt{tanh}(\mathbf{W}_v[\mathbf{v}_t; \mathbf{s}_t]) \quad (3)$$

They are combined in a weighted sum to obtain an attentional decoder state $\tilde{\mathbf{s}}_\mathbf{t}$, where weights are calculated by a sparse gate vector $\mathbf{p_t} = [p_0, p_1] \in \mathbb{R}^2$:

$$\mathbf{p}_t = \texttt{sparsemax}(\mathbf{W}_g[\mathbf{u}_t; \mathbf{v}_t; \mathbf{s}_t]) \quad (4)$$

$$\tilde{\mathbf{s}}_t = p_0\tilde{\mathbf{s}}_{tu} + p_1\tilde{\mathbf{s}}_{tv} \quad (5)$$

The attentional decoder state is fed into a sparse prediction layer. For the input feeding, the input to decoder comprises the predicted symbol embedding and gated attention vector c_t:

$$\mathbf{c}_t = p_0\mathbf{u}_t + p_1\mathbf{v}_t \quad (6)$$

The two-headed gate mechanism 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.

Integrating Self-Att^{sub} We depart from the existing character-level solution in that we assume that the input to the model - a (lemma, MSD) pair - is complemented with the segmentation of lemma into subwords.[7] We obtain an extra subword representation of lemma $\mathbf{H}^{subw}$ by averaging lemma representation vectors in $\mathbf{H}^u$ spanning characters within each subword. Besides the attention heads $\mathbf{u}_t$ and $\mathbf{u}_t$ computed at each generation step, we construct an additional attention head vector $\mathbf{m}$ which is computed once before the decoding stage. It is constructed by scoring the sequence of lemma subword representations in $\mathbf{H}^{subw}$ with a query vector $\mathbf{q}^{pos}$ corresponding to the encoding of the lemma's POS tag. This encoding is obtained by selecting a vector in MSD representation $\mathbf{H}^v$ corresponding to the position of the POS tag (e.g, POS tag *V* (verb) is in the first position in MSD *V;IND;PRS;3;PL*).

To integrate subword-level attention head $\mathbf{m}$ in the baseline system, we modify the gate layer in Eq. 4:

$$\mathbf{p}_t = \texttt{sparsemax}(\mathbf{W}_g[\mathbf{m}; \mathbf{u}_t; \mathbf{v}_t; \mathbf{s}_t]); \quad (7)$$

In this way, the gate mechanism (and decoding) is expected to be informed with a signal for inflection class selection when such signal can be attributed to specific character spans (subwords) in the lemma. The attention over subwords is static and shared across target positions, aiming to separate the signal of the class assignment it conveys from local character transformations, given this assignment.

4 Pattern extraction

To extract linguistic rules from the trained model Cross-Att^{ch}Self-Att^{sub} , we represent its knowledge of inflection as a database. To this end, we take predictions of the model on a dataset, which can be the original task data for model training or a dataset collected to study a specific inflection phenomenon. Then, for each example in the dataset, we populate the knowledge database with the example itself and two patterns, which are extracted from the learned attention weights. The first one is a transformation pattern (§4.1) obtained by applying our pattern extraction method to learned cross-attention weights (Cross-Att^{ch} component). This method can be applied to any inflection model embedded into encoder-decoder paradigm with attention. The second pattern is over the lemma subwords (§4.2) which is obtained from attention weights of the novel Self-Att^{sub} component. Finally, we explain how populated in this way knowledge database can be queried to study inflection phenomena (§4.3).

4.1 Cross-Att^{ch} Transformation Patterns

This method maps each example *(lemma, MSD)* $\rightarrow$ *inflected form* to a transformation pattern of a

[6] sparsemax activations (Martins and Astudillo, 2016) serve as a sparse (and therefore, more interpretable) alternative to a commonly used attention function - softmax. The latter yields dense attention weights: all elements in the input always make at least a small contribution to the decision.

[7] Such representation can be obtained with any off-the-shelf segmentation algorithms, e.g. BPE (Sennrich et al., 2016a) or Morfessor (Smit et al., 2014).

Input:	
$X = \underset{123456789}{scolorire}$	$F = \underset{1}{V}\ \underset{2}{IND}\ \underset{3}{PRS}\ \underset{4}{3}\ \underset{5}{PL}$
$Y = \underset{123456789101112}{scolorisc\,o\,n\,o}$	A^X and A^F as in Figure 1
Step 1: Transform att. weights into 'salient' alignments	
$A = [\underset{copy}{X_1 \cdots X_7}, F_4, F_3, F_2, F_4, F_4]$	
Step 2: Inverse A and group pred. steps by gen. type	
$X_1 \rightarrow \{c : [1]\}$	$F_4 \rightarrow \{f4 : [8, 11, 12]\}$
$\cdots$	$F_3 \rightarrow \{f3 : [9]\}$
$X_7 \rightarrow \{c : [7]\}$	$F_2 \rightarrow \{f2 : [10]\}$
Step 3: Replace char. in X and Y with indexed gen. type	
$P^{tr}(X) = \underset{7}{\mathrm{c}^1 \cdots \mathrm{c}^1}$ *re*	
$P^{tr}(Y) = \underset{7}{\mathrm{c}^1 \cdots \mathrm{c}^1}\ \mathrm{f4}^1\ \mathrm{f3}^1\ \mathrm{f2}^1\ \mathrm{f4}^2\ \mathrm{f4}^2$	
Step 4: Collapse adjacent symbols	
$P^{tr}(X)$=c^1 *re*	
$P^{tr}(Y) = \mathrm{c}^1\ \mathrm{f4}^1\ \mathrm{f3}^1\ \mathrm{f2}^1\ \mathrm{f4}^2$	
Output:	
c^1 *re* $\rightarrow \mathrm{c}^1\ \mathrm{f4}^1\ \mathrm{f3}^1\ \mathrm{f2}^1\ \mathrm{f4}^2$	
($\mathrm{c}^1 \leftrightarrow$ *scolori*, $f4^1 \leftrightarrow$ *s*, $f3^1 \leftrightarrow$ *c*, $f2^1 \leftrightarrow$ *o*, $f4^2 \leftrightarrow$ *no*)	

Table 2: Illustration of Cross-Att$^{\text{ch}}$ Transformation Patterns algorithm applied to the example in Fig. 1.

form $P^{tr}(lemma) \rightarrow P^{tr}$*(inflected form)*. Formally, the input to the algorithm is a lemma $X = x_1, \ldots x_n$, MSD $F = f_1 \ldots f_l$, predicted target form $Y = y_1 \ldots y_m$ and cross-attention weights over lemma characters $A^X = a_1^X \ldots a_m^X$, $a_i^X \in \mathbb{R}^n$ and over MSD tags $A^F = a_1^F \ldots a_m^F$, $a_i^F \in \mathbb{R}^l$.[8] The output is a string of a form $P^{dec}(X) \rightarrow P^{dec}(Y)$ where the constructed pattern representation P^{dec} for the lemma and target are built through the following steps (shown in Table 2 for our example in Fig. 1):

Step 1. Transform input attention weights A^x and A^f into "salient" alignments A: Each component a_j of salient alignments $A = a_1 \ldots a_m$ is a set of input positions (in lemma X and/or MSD F) that provide the most significant contributions to predicting a character in Y at position j. We denote positions by capitalized symbols, i.e. $F1$ for position 1 in F, to reflect the difference between the position's index and value. Salient alignments are built by applying a filtering function ϕ to attention weights at each predicted position: $\phi : [a_j^X; a_j^F] \rightarrow a_j$. In the following, we illustrate how our algorithm works for the simplest choice of the filtering function, max-pooling, which simply selects one input position with the highest attention weight.[9] In our running example, this strategy results in only one element for each component a_j: e.g. $a_7 = X_7$.

Step 2. Inverse mapping A and group prediction steps by generation type: By inverting salient alignments, we construct a mapping from input positions to prediction steps grouped by a symbol corresponding to generation type. The latter is identified for each alignment a_j by the type of input: we denote generation from the lemma's characters ($a_j = X_i$) by symbol g, whereas that from a tag ($a_j = F_k$) is denoted by indexed symbol fk. The special case of copying a character from the lemma, i.e. $a_j = X_i$ and $x_i = y_j$, is denoted by symbol c. Thus, a position in F can be mapped to only one group of prediction steps (the type of generation is unique and defined by the tag's position), whereas that in X can be mapped to up to two groups, g and c. Some input positions might be absent in the constructed mapping, if not present in salient alignments, e.g. X_9 in Table 2.

Step 3. Replace characters in X and Y with indexed generation type symbols: We index (in the order of input positions) triples of salient alignments (input position, generation step, generation type) identified in the previous step. Then, we construct patterns of lemma and inflected form, by replacing characters at aligned positions with an indexed value of the generation type symbol, e.g. $(c, X_j, Y_k) \rightarrow index; x_j \rightarrow c^{index}; y_k \rightarrow c^{index}$. In X, this can result in an aggregated symbol, e.g. replacing X_i with $c^{1;2}; g^1$ means that position X_i is aligned to three target positions, two of which are generated by copying x_i. As illustrated in our running example, we use the same index value in two special cases: a) a whole target substring was copied, and b) a whole target substring was generated by the same tag. We keep the track of symbolic mappings from characters to indexed generation symbols that replace them.

Step 4. Collapse adjacent symbols: Scan representations $P^{tr}(X)$ and $P^{tr}(Y)$, built at the previous step and iteratively collapse adjacent symbols of the same value. At the same time, we update

[8] We assume that the sum of weights in a combined vector [A^X;A^F] is 1. In Cross-Att$^{\text{ch}}$Self-Att$^{\text{sub}}$ model this is achieved by scaling cross-attention weights for the lemma and MSD with corresponding gate values. Another way, typical for neural inflection models of encoder-decoder class, is to run cross-attention over a concatenation of the lemma's characters and MSD's tags.

[9] `sparsemax` activations provide another choice to filtering function by keeping the input positions corresponding to nonzero attention weights. In our example, this would result in e.g. $a_7 = \{X_7, F_2\}$. We refer to a more general form of the algorithm covering such case in the Appendix.

the symbolic mapping: if two adjacent symbols are collapsed, we replace their string mappings with a single mapping from the strings concatenation to the generation symbol.

The idea behind the inverse mapping and indexing in steps 2 and 3 is to ensure a unique way of indexing generation symbols across all data pairs. The indices themselves are essential to keep a one-to-one mapping from substrings to the generation symbols they are replaced with. Both factors come into play when we query the knowledge database for an inflection phenomenon (§4.3).

4.2 Self-Att^{sub} Lemma Patterns

This algorithm takes as an input a data example (X, Y, F), along with a segmented lemma representation $S(X) = s_1 \dots s_p$ and learned self-attention weights over lemma's subwords: $a^{S(X)} \in \mathbb{R}^p$. The output is a pattern for salient subwords in lemma $P^l(X)$ which is built with a similar procedure as described above where indexing steps 2 and 3 are skipped.

First, we transform self-attention weights $a^{S(X)}$ into salient alignments a^{enc} by applying a filtering function: $\phi : a^{S(X)} \rightarrow a$, thereby identifying a set of subword positions with the most significant contribution to the overall generation process (any type of filtering function described in the previous subsection can be applied). Afterward, we replace all subwords in the input lemma at nonsalient positions, $S_j \notin a$, with a dedicated symbol, e.g. asterisk *. Finally, we iteratively merge adjacent asterisk symbols to obtain a more general pattern. To illustrate with our running example, given a segmented representation of lemma $S(X) = s|col|or|i|re$ and salient alignments $a = \{S4, S5\}$, obtained by filtering input positions with nonzero self-attention weights, the resulting pattern for salient subwords in the lemma is $P^l(X) = *ire$.

4.3 Querying Patterns

As a result of applying the previous two methods, each data example (X, Y, F), along with segmented lemma representation $S(X)$ and learned attention weights (A^X, A^F, $a^{S(X)}$), can be mapped into two items: Cross-Att^{ch} transformation pattern $P^{tr}(X) \rightarrow P^{tr}(Y)$ and Self-Att^{sub} pattern for salient subwords in lemma $P^l(X)$. The data examples along with the extracted patterns are stored in a knowledge database. To systematically study how the neural model handles a specific linguistic phenomenon of interest, the database can be queried, for patterns and examples, with a phenomenon's formalization in a form of regular expressions applied to the lemma, inflected form or MSD. Selected with a query examples are then grouped by their patterns (either transformation or lemma ones) resulting in each group representing an induced linguistic rule for the phenomenon.

At this stage, to make the patterns more readable, we perform an unmasking operation within each group: if a particular symbol is used to substitute one substring that is the same for all examples within a group, we replace the symbol back with this substring. For instance, if the pattern from our example c^1 *re* $\rightarrow$ $\mathrm{c}^1\,\mathrm{f4}^1\,\mathrm{f3}^1\,\mathrm{f2}^1\,\mathrm{f4}^2$ represents one such group, and symbol $\mathrm{f4}^2$ is used to substitute only one string *no*, which is the same across all data points in the group, we can unmask the string, to obtain a pattern c^1 *re* $\rightarrow$ $\mathrm{c}^1\,\mathrm{f4}^1\,\mathrm{f3}^1\,\mathrm{f2}^1$ *no*.

5 Experiments and Results

We perform three case studies, introduced in §2.1, to demonstrate how our framework allows querying patterns learned by an inflection neural model. The goal of our experiments is to assess how well the extracted patterns correspond to known inflection rules. To see whether our modifications to the inflection model affect its performance, we check the inflection accuracy on the analyzed languages and compare it with the original character-level model.

We use data from SIGMORPHON shared task: 2018 edition for Italian and Finnish (10K/1K/1K examples in train/development/test data), and 2020 edition for Tagalog (1,870/236/478). For each language, we train Cross-Att^{ch}Self-Att^{sub} model with batch size 4, beam size 1 and other hyperparameters as reported in Peters and Martins (2019). To produce segmented lemma input, we use the BPE method (Gage, 1994; Sennrich et al., 2016b) with 1K merges on a token list (100K examples) extracted from WikipediaDumps articles.[10].

Using model's predictions on the concatenation of train, development and test set, we query Cross-Att^{ch} and Self-Att^{sub} patterns As a filtering function, we keep only nonzero weights for Self-Att^{sub} patterns, whereas we choose max-pooling for Cross-Att^{ch} ones, as on average sparse activations assign

[10]We use archives of the name format enwiki-20190920-pages-articles.xml.bz2 from https://ftp.acc.umu.se/mirror/wikimedia.org/dumps/

nonzero weight to one input feature. To systematically examine whether the classes of patterns extracted are correct and cover the data adequately, we report two metrics, namely a) number of examples selected with a query and how many of them are grouped by each pattern, and b) model accuracy (correct predictions) with respect to the number of examples per selection with a query and per group pattern.

Cross-Att^{ch}: Transformation Patterns For each language, we define specific queries: 3rd person plural present tense for Italian (MSD=*V;IND;PRS;3;PL*), 3rd person plural present positive imperative for Finnish (MSD=*V;ACT;PRS;POS;IMP;3;PL*) and imperfective aspect with agent semantic role for Tagalog (MSD=*V;IPFV;AGFOC*). The choice of MSDs is rather arbitrary: for illustrative purposes we select grammatical categories that contain enough examples to represent form variation of the corresponding morpheme. Table 3 present the extracted Cross-Att^{ch} patterns. For each query, we show patterns that group at least 5% of the examples selected with a query. The patterns are sorted by their number of examples in a decreasing order. For each presented pattern, we show an example mapped to this pattern and symbol mapping information. The latter lists, for each symbol in the pattern, all substrings mapped to this symbol along with their frequencies (within a group), if the number of distinct substrings is less than five elements. Otherwise, we show average length (≈) of substrings mapped to this symbol, or exact length (=), if it is the same for all of them. These symbol mappings also include bijection cases (↔) that were unmasked after grouping examples (as described in §4.3)

We observe that in all three cases, the patterns recover inflection morphemes listed in grammars for studied grammatical categories as well as their form variation. For Finnish, the model correctly identifies morpheme *-koot* as well as its variant *-kööt* because of vowel harmony. Additionally, the patterns (3) and (4) display morphophonological processes on morphemes boundaries: if a stem ends with *-d* or *-l*, this ending is removed from the final form. In Italian, the morphemes for all three inflection classes are present in the patterns: *-ano* (*-are* class) *-ono* (*-ere* class) and *-scono* (*-ire* class). Besides, the separation of reflexive ending *si* is visible (pattern (2)) for the *-are* class. Tagalog patterns detect two frequent inflection classes corresponding to so-called *um*-verbs and *nag*-verbs. Analyzing symbolic mappings, we conclude that the pattern (1) encodes prefixation (the most frequent prefix in

Transformation Patterns	No. of/Acc
Finnish	
Q: MSD=*V;ACT;PRS;POS;IMP;3;PL*	46/0.91
(1) $\mathbf{c^1}$ **a** → $\mathbf{c^1}$ **k oo t** *karsastaa* : *karsastakoot* $\lvert\mathbf{c^1}\rvert \approx 7.1$; $\mathbf{f6^1} \leftrightarrow k$; $\mathbf{f5^1} \leftrightarrow oo$; $\mathbf{f7^1} \leftrightarrow t$	23/1.00
(2) $\mathbf{c^1}$ **ä** → $\mathbf{c^1}$ **k öö t** *mylviä* : *mylvikööt* $\lvert\mathbf{c^1}\rvert \approx 7.4$; $\mathbf{f6^1} \leftrightarrow k$; $\mathbf{f5^1} \leftrightarrow öö$; $\mathbf{f7^1} \leftrightarrow t$	7/1.00
(3) $\mathbf{c^1}$ **d a** → $\mathbf{c^1}$ **k oo t** *promovoida* : *promovoikoot* $\lvert\mathbf{c^1}\rvert \approx 8.4$; $\mathbf{f6^1} \leftrightarrow k$; $\mathbf{f5^1} \leftrightarrow oo$; $\mathbf{f7^1} \leftrightarrow t$	5/1.00
(4) $\mathbf{c^1}$ **l a** → $\mathbf{c^1}$ **k oo t** *aaltoilla* : *aaltoilkoot* $\lvert\mathbf{c^1}\rvert \approx 7.7$; $\mathbf{f6^1} \leftrightarrow k$; $\mathbf{f5^1} \leftrightarrow oo$; $\mathbf{f7^1} \leftrightarrow t$	3/1.00
Italian	
Q: MSD=*V;IND;PRS;3;PL*	255/0.99
(1) $\mathbf{c^1}$ **a r e** → $\mathbf{c^1}$ **a n o** *zampicare* : *zampicano* $\lvert\mathbf{c^1}\rvert \approx 6.7$; $\mathbf{f3^1} \leftrightarrow a$; $\mathbf{f4^1} \leftrightarrow n$; $\mathbf{f5^1} \leftrightarrow o$	149/1.00
(2) $\mathbf{c^1}$ **a r s i** → **si** $\mathbf{c^1}$ **a n o** *impaperarsi* : *si impaperano* $\lvert\mathbf{c^1}\rvert \approx 6.8$; $\mathbf{f1^1} \leftrightarrow si$; $\mathbf{f3^1} \leftrightarrow a$; $\mathbf{f4^1} \leftrightarrow n$ $\mathbf{f5^1} \leftrightarrow o$	40/1.00
(3) $\mathbf{c^1}$ **e r e** → $\mathbf{c^1}$ **o n o** *rirompere* : *rirompono* $\lvert\mathbf{c^1}\rvert \approx 7.1$; $\mathbf{f2^1} \leftrightarrow o$; $\mathbf{f4^1} \leftrightarrow n$; $\mathbf{f5^1} \leftrightarrow o$	23/1.00
(4) $\mathbf{c^1}$ **r e** → $\mathbf{c^1}$ **s c o n o** *scolorire* : *scolorisconо* $\lvert\mathbf{c^1}\rvert \approx 7.3$; $\mathbf{f4^1} \leftrightarrow s$; $\mathbf{f5^1} \leftrightarrow c$; $\mathbf{f2^1} \leftrightarrow o$ $\mathbf{f4^2} \leftrightarrow n$; $\mathbf{f5^2} \leftrightarrow o$	16/1.00
Tagalog	
Q: MSD=*V;IPFV;AGFOC*	377/0.86
(1) $\mathbf{c^{1;2}}$ $\mathbf{c^{3;4}}$ $\mathbf{c^5}$ → **n** $\mathbf{f3^1}$ $\mathbf{c^1}$ $\mathbf{c^3}$ $\mathbf{c^2}$ $\mathbf{c^4}$ $\mathbf{c^5}$ *paalam* : *nagpapaalam* $\lvert\mathbf{c^1}\rvert$=1; $\lvert\mathbf{c^2}\rvert$=1; $\lvert\mathbf{c^{1;2}}\rvert$=1; $\lvert\mathbf{c^{3;4}}\rvert$=1 $\mathbf{c^3}$: {*a* (82), *u* (34), *i* (23), *e* (3)} $\mathbf{c^4}$: {*a* (82), *u* (34), *i* (23), *e* (3)} $\mathbf{f3^1}$:{*ag* (131), *a* (8), *ang* (1), *an* (2)} $\mathbf{f2^1} \leftrightarrow n$; $\lvert\mathbf{c^5}\rvert \approx 3.3$	142/0.90
(2) $\mathbf{c^1}$ $\mathbf{c^{2;3}}$ $\mathbf{c^4}$ → $\mathbf{f2^1}$ $\mathbf{f3^1}$ $\mathbf{c^2}$ $\mathbf{c^1}$ $\mathbf{c^3}$ $\mathbf{c^4}$ *hiram* : *humihiram* $\lvert\mathbf{f2^1}\rvert$=1; $\lvert\mathbf{c^1}\rvert$=1;$\lvert\mathbf{c^{2;3}}\rvert$=1; $\lvert\mathbf{c^4}\rvert \approx 3.1$ $\mathbf{f3^1}$: {*um* (71), *am* (3), *k* (1), *as* (1), *an* (2)} $\mathbf{c^2}$: {*i* (24), *a* (35), *u* (17), *o* (2)} $\mathbf{c^3}$: {*i* (24), *a* (35), *u* (17), *o* (2)}	78/0.86

Table 3: Cross-Att^{ch} transformation patterns. *Q* is a query regular expression. The number of examples (*No of*) and accuracy (*Acc*) are shown per selection with a query and per group pattern.

this group is *nag*) with subsequent copying of the first syllable and copying of the full lemma string. The pattern (2) expresses reduplication of the first consonant (which interestingly gets aligned to a tag rather than to the first character of the lemma), generating infix (the most frequent infix in this group is *um*), copying the second character of the lemma (vowel, as seen from the mapping statistics), and then copying of full lemma string.

Self-Att^{sub} : Lemma Patterns We analyze Self-Att^{sub} lemma patterns to determine whether our model uses indeed subword segments when choosing the specific variant of a morpheme. Concretely, we use regular expressions on the target form to select examples corresponding to a specific form of morpheme, identified above with transformation patterns. Then, we map selected examples to their Self-Att^{sub} patterns. Table 4 presents the queries and extracted patterns.[11] For each query, we list the most frequent patterns (sorted by frequency in a decreasing order) along with one segmented lemma example mapped to the pattern.[12] The segments of lemma examples, identified as salient (and presented in the patterns) are highlighted in bold.

We conclude that the subword regions identified by Self-Att^{sub} patterns conform to a great extent to triggers of morpheme form variation listed in grammars. We note that although the regions for finding such clues (when they are phonological or lexical, and frequent) look plausible, their form is influenced by the results of BPE segmentation and may not be perfectly aligned with grammars. For example, Italian patterns show that the model's focus is on the endings of lemmas for all three classes. In case of reflexive verbs, where reflexive ending *-si* tends to be separated into a separate subword by BPE, the model correctly places focus on a more informative penultimate segment. The patterns extracted for Finnish, display the grammar rules too: the focus on the lemma endings *-aa* and *-ua* for the first group, and *-ää*/*-ä* for the second group, points directly to the harmony of back and front vowels, respectively. The model does not search for the clues in the vowel patterns of the stem but chooses a smart strategy to focus directly on the inflection endings for lemmas: they are frequent and already agree with the vowels found elsewhere in the stem to the left.

Lemma Patterns	No. of/Acc
Finnish	
Q: gold target=*koot & MSD=msd_fin	37/0.97
aa** (*kar*|*sa*|*st*|aa***)	8/1.00
ua** (*ku*|*or*|*ett*|ua***)	5/1.00
Q: gold target=*kööt & MSD=msd_fin	9/1.00
ää** (*jä*|*n*|*ist*|ää***)	3/1.00
ä** (*v*|*et*|*ele*|*hti*|ä***)	3/1.00
Italian	
Q: gold target=*scono & MSD=msd_it	23/1.00
re** (*in*| *z*| *o*| *ti*|*chi*| ***re)	9/1.00
ire** (*s*| *col*| *or*| ***ire)	7/1.00
ir (*re*| *in*| *ser*| ***ir***| *si*)	6/1.00
Q: gold target=*ano & MSD=msd_it	189/1.00
are** (*z*| *am*| *pic*| ***are)	149/1.00
arsi** (*im*| *pa*| *per*| ***arsi)	26/1.00
car (*ri*| *mb*| *ec*| ***car***| *si*)	3/1.00
Q: gold target=*ono & !(*scono) & MSD=msd_it	41/0.95
ere** (*ri*| *otten*| ***ere)	19/0.95
dere** (*te*| *le*| *ve*| ***dere)	10/1.0
ger (*cos*| *par*| ***ger***| *si*)	3/1.00

Table 4: Self-Att^{sub} Lemma Patterns. *Q* is a query regular expression, *msd_fin* is *V;ACT;PRS;POS;IMP;3;PL*, and *msd_it* is *V;IND;PRS;3;PL*. The number of examples (*No of*) and accuracy (*Acc*) are shown per selection with a query and per group pattern.

Self-Att^{sub} : Performance Impact We evaluate the impact of the novel Self-Att^{sub} component by comparing the performance of Cross-Att^{ch}Self-Att^{sub} with that of the baseline model, Cross-Att^{ch} . For reference, we include the results of a) the hard monotonic attention (HMA) system of Wu and Cotterell (2019) which currently holds as the state-of-the-art on the reinflection task by rerunning their code; b) a variant of our system, Cross-Att^{ch}Self-Att^{ch} where the encoder attention module is run over the characters of the lemma, instead of subwords. The latter corresponds to a limiting case of lemma segmentation where each character is a segment. We report accuracy and edit distance on the test set in Table 5. Additionally, we provide information on the number of trained parameters for each model. The number of parameters for Cross-Att^{ch}Self-Att^{sub} model is the same as for its character variant Cross-Att^{ch}Self-Att^{ch} . The difference in the number of parameters across the languages is due to the variation of their character vocabulary sizes.

[11] For Tagalog, we note that we do not find any frequent patterns for e.g. a query "gold target=*nag**", which is in line with no explicit criteria for inflection class assignment in this language.

[12] We refer to the Appendix, Tables 6-7 for the full list of extracted patterns.

	Baseline	Our model	Comparison	
	Cross-Att^{ch}	Cross-Att^{ch}Self-Att^{sub}	Cross-Att^{ch}Self-Att^{ch}	HMA
Italian	95.40 (0.09) [*1,742K*]	96.70 (0.09) [*1,783K*]	**97.40** (**0.06**) [*1,783K*]	96.80 (0.29) [*8,647K*]
Finnish	93.80 (0.13) [*1,758K*]	**94.40** (**0.09**) [*1,798K*]	93.60 (0.12) [*1,798K*]	93.90 (0.13) [*8,709K*]
Tagalog	65.75 (1.21) [*1,739K*]	**69.98** (**0.92**) [*1,780K*]	66.81 (1.00) [*1,780K*]	63.39 (1.53) [*8,623K*]

Table 5: Accuracy (and edit distance) on the test set. The number of model parameters is given in squared brackets.

We observe, that the Cross-Att^{ch}Self-Att^{sub} model shows systematic improvements across all three languages over the baseline and reference models. Regarding the level of segmentation, the Cross-Att^{ch}Self-Att^{ch} system achieves higher results on Italian, where indeed, class variation can be associated with a certain character in a certain position. In terms of the number of trained parameters, the improvements due to the Self-Att^{sub} component are achieved by only adding a relatively small number of extra parameters compared with the baseline model, Cross-Att^{ch} . We also note that the performance of our systems is higher or on par with the state-of-the art model HMA, whereas the latter has an on-average sevenfold increase in the number of parameters in comparison with to that of Cross-Att^{ch}Self-Att^{sub} and Cross-Att^{ch}Self-Att^{ch} .

6 Discussion and Future Work

In the following, we discuss our proposed methodology in terms of two aspects, namely, interpretability for inflection (in terms of typological parameters) and ideas for performance improvement.

Interpretability for Inflection In terms of the typological parameter of **fusion**, the results of our experiments illustrate that our Cross-Att^{ch} pattern approach can effectively extract rules for concatenative morpheme forms as well as reduplication processes. What is beyond, at the moment, are nonlinear processes that are not always visible in orthography, e.g. tonal changes and internal stem changes. The latter, for example, is demonstrated by root and pattern morphology in Arabic and Hebrew, for which standard orthographies do not indicate most vowels.

Regarding **flexivity**, our Self-Att^{sub} pattern method can identify phonological (visible in orthography) as well as lexical triggers to the variation of inflection morpheme's form. However, the case of suppletive forms (English *go*→ *went*) would not be identifiable in patterns. Although suppletive cases are likely to be fairly rare in terms of word types, they seem to be only maintained in high-frequency words (Bybee, 1985). Therefore, although affecting only a small number of words, suppletion might be visible in patterns when studied together with word frequency (which is, at the moment, not possible because of the current practices for building inflection generation datasets).

The parameter of **exponence** encodes the extent to which single morphemes express multiple morphosyntactic features. For the class of neural models currently used for inflection generation, it is not possible to see a clear correspondence between the meaning assigned by humans and the model: as we see from Fig. 1 which illustrates polyexponence in Italian inflection, the model assigns separate characters of inflection morpheme *-scono* to different tags, whereas for humans, it is hard to break down this morpheme into smaller meaningful parts.[13]

Performance Future work can evaluate the impact of Self-Att^{sub} in combination with frequently used induction biases[14], as well as transformers paradigm, which recently proved to be effective on the task. (Vylomova et al., 2020).

7 Conclusion

We propose a novel approach for interpreting neural inflection models by extracting patterns from attention weights. To enhance the interpretability of this class of models, we design a linguistically motivated attention component over subwords that leads to a systematic performance improvement. Our experiments with linguistic rules induction illustrate the great potential of our methodology for linguistic research scaled to diverse typology.

Acknowledgments

This work has been partially supported by the SNSF grant no. 176305.

[13]This challenge is exemplified by the input features identity problem in attention weights (Brunner et al., 2019).

[14]For example, hard monotonic attention which is highly effective for Indo-European languages with concatentive morphology.

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.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Emily M Bender. 2011. On achieving and evaluating language-independence in nlp. *Linguistic Issues in Language Technology*, 6(3):1–26.

Balthasar Bickel and Johanna Nichols. 2007. Inflectional morphology. *Language typology and syntactic description*, 3(2):169–240.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2019. On identifiability in transformers. In *International Conference on Learning Representations*.

Joan L Bybee. 1985. *Morphology: A study of the relation between meaning and form*, volume 9. John Benjamins Publishing.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine 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. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. Association for Computational Linguistics.

Matthew S Dryer and Martin Haspelmath. 2013. The world atlas of language structures online.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? *arXiv preprint arXiv:2004.03685*.

Zachary C. Lipton. 2018. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43.

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.

D Christopher Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences of the United States of America*.

André Martinet. 1967. Eléments de linguistique générale. *Colin*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. volume 48 of *Proceedings of Machine Learning Research*, pages 1614–1623, New York, New York, USA. PMLR.

Ben Peters and André FT 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.

Martin Schrimpf, Idan Blank, Greta Tuckute, Carina Kauf, Eghbal A. Hosseini, Nancy Kanwisher, Joshua Tenenbaum, and Evelina Fedorenko. 2020. Artificial neural networks accurately predict language processing in the brain. *bioRxiv*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Timothy Shopen. 1985. *Language Typology and Syntactic Description: Volume 3*, volume 3. Cambridge University Press.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden. Association for Computational Linguistics.

Ekaterina Vylomova, Jennifer White, Elizabeth Salesky, Sabrina J Mielke, Shijie Wu, Edoardo Ponti, Rowan Hall Maudslay, Ran Zmigrod, Josef Valvoda, Svetlana Toldova, et al. 2020. Sigmorphon 2020 shared task 0: Typologically diverse morphological inflection. *arXiv preprint arXiv:2006.11572*.

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.

A Cross-Att^{ch} Transformation Patterns Algorithm

In this Section, we formalize Steps 2 and 3 of the algorithm for pattern extraction from character-level cross-attention weights presented in §5.

Algorithm 1: (Step 2) Inverse salient alignments mapping A and group prediction steps by generation type

Inputs:
$X \leftarrow [x_1 \dots x_n]$; // Lemma
$F \leftarrow [f_1 \dots f_n]$; // MSD
$Y \leftarrow [y_1 \dots y_m]$; // Target
$A \leftarrow [a_1 \dots a_m]$; // Salient alignments
Init: $X\_pos\_map = \{\}$, $F\_pos\_map = \{\}$ will store salient mappings from input positions to prediction steps, grouped by generation type.
for a_j in A:
  for P in a_j: ; // salient alignments to y_j
    if $P == X_i$: ; // aligned to lemma
      if $x_i == y_j$: ; // copy
        add j to $X\_pos\_map[X_i][c]$
      else:
        add j to $X\_pos\_map[X_i][g]$
    else ($P == F_k$): ; // aligned to tag
      add j to $F\_pos\_map[F_k]$
Outputs: X_pos_map, F_pos_map

Algorithm 2: (Step 3.1) Replace characters in Y with indexed generation type symbols using salient alignments to F

Inputs: F_pos_map; $\tilde{Y}$=copy(Y)
Init: $f2prev\_target = \{\}$; $f2ind = \{\}$
for F_k in F:
  if F_k in F_pos_map:
    for j in $F\_pos\_map[F_k]$: ; // Y indexes
      if f_k not in ($f2prev\_target$)):
        $f2ind[f_k]$=1; ; // If nothing was aligned yet to f_k, we create an index
        if $\tilde{y}_j$ is not replaced:
          $\tilde{y}_j \rightarrow f_k^1$
        else:
          $\tilde{y}_j$+= f_k^1
      else:
        if ($f2prev\_target[f_k] + 1$) != j:
          $f2ind[f_k]$+=1 ; // If something was aligned to f_k, check the last target step saved. Only increment it if it's not the same
        index=$f2ind[f_k]$
        if $\tilde{y}_j$ is not replaced:
          $\tilde{y}_j \rightarrow f_k^{index}$
        else:
          $\tilde{y}_j$+= f_k^{index}
Outputs: $P^{tr}(Y) = \tilde{Y}$

Algorithm 3: (Step 3.2) Replace characters in X and Y with indexed generation type symbols using salient alignments to X

Inputs: X_pos_map, $P^{tr}(Y)$
Init: $c_{index} = 1$; $g_{index} = 1$; $\tilde{X}$=copy(X); $\tilde{Y} = P^{dec}(Y)$
for X_i in X:
  if X_i in X_pos_map:
    $c(X_i) = X\_op\_map[X_i][c]$
    $g(X_i) = X\_op\_map[X_i][g]$
    if $c(X_i) == [j]$ and $x_i == y_j$ and $g(X_i) == \emptyset$: ; // X_i is 1-to-1 copy
      if X_{i-1} is not adjacent 1-to-1 copy:
        $c_{index} += 1$
      $\tilde{x}_i = C^{c_{index}}$; $\tilde{y}_j = C^{c_{index}}$
    else:
      if $c(X_i)! = \emptyset$ and $g(X_i)! = \emptyset$:
        $mask =$''; $full\_index = []$
        for k in $c(X_i)$:
          c_{index}+=1
          add c_{index} to $full\_index$
          if $\tilde{y}_k$ is not replaced:
            $\tilde{y}_k \rightarrow C^{c\_index}$
          else:
            $\tilde{y}_k$+= C^{c_index}
        $mask$+='C^{full_index},
        $full\_index = []$
        for k in $g(X_i)$:
          g_{index}+=1
          add g_{index} to $full\_index$
          if $\tilde{y}_k$ is not replaced:
            $\tilde{y}_k \rightarrow G^{g\_index}$
          else:
            $\tilde{y}_k$+=G^{g_index}
        $mask$+='G^{full_index},
        $\tilde{x}_j \rightarrow mask$
Outputs: $P^{tr}(X) = \tilde{X}$, $P^{tr}(Y) = \tilde{Y}$

B Self-Att^{sub} Lemma Patterns

Query	No. of/Acc	Patterns
gold target=*scono & MSD=msd_it	23/1.00	*re: 9/1.0 (in| z| o| ti| chi| re) *ire: 7/1.0 (s| col| or| ire) *ir: 6/1.0 (re| in| ser| ir| si) *cir*: 1/1.0 (in| fer| o| cir |si)
gold target=*ano & MSD=msd_it	189/1.00	*are: 149/1.0 (z| am| pic| are) *arsi: 26//1.0 (im| pa| per| arsi) *car*:3/1.0 (ri|mb|ec|car|si) *izzarsi:2/1.0 (dest|abil|izz|arsi) *iarsi:2/1.0 (di|lan|i|arsi) *par*:2/1.0 (dis|col|par|si) *ciarsi:1/1.0 (au|to|den|un|ci|arsi) *mar*:1/1.0 (in|for|mar|si) *rarsi:1/1.0 (gi|ost|r|arsi) *itarsi:1/1.0 (ri|abil|it|arsi) *quar*:1/1.0 (sci|ac|quar|si)
gold target=*ono & !(*scono) & MSD=msd_it	41/0.95	*ere: 18/0.95 (ri| otten| ere) *dere: 10/1.0 (te| le| ve| dere) *ger*: 3/1.0 (cos| par| ger| si) *re:3/1.0 (servi|re) *e:1/0.0 (ri|ro|m|per|e) *ir*:1/1.0 (1908:s|ent|ir|si) *si:1/1.0 (es|p|or|si) *ire:1/1.0 (ri|di|ven|ire) *er*:1/1.0 (r|aggi|ung|er|si) *mer*:1/1.0 (ass|u|mer|si)

Table 6: Italian Self-Att^{sub} Patterns. MSD query *msd_it* is *V;IND;PRS;3;PL*. Number of examples (*No of*) and accuracy (*Acc*) are shown per selection with query and per group pattern. For each query, we list all extracted lemma patterns (sorted by frequency in a decreasing order) along with one segmented lemma example (in parentheses) mapped to the pattern.

Query	No. of/Acc	Patterns
gold target=*koot & MSD=msd_fin	37/0.97	*aa:8/1.0 (kar|sa |st |aa) *ua:5/1.0 (ku |or |ett |ua) *id*:5/1.0 (pro |mo |vo |id |a) *a:4/1.0 (pu |r |je |hti |a) *ta:4/1.0 (sk |r |uud |a |ta) *taa:4/1.0 (jo |kel |taa) *illa:2/1.0 (aal |to |illa) *ella:2/0.5 (n |ar |a |hd |ella) *ttaa:1/1.0 (ha |h |mo |ttaa) *sia:1/1.0 (har |sia), *ista:1/1.0 (li |i |pa |ista)
gold target=*kööt & MSD=msd_fin	9/1.00	*ä:3/1.0 (v |et |ele |hti |ä) *ää:3/1.0 (jä |n |ist |ää) *tä:2/1.0 (kä |pä |tä) *tää:1/1.0 (hy |mä |h |ää)

Table 7: Finnish Self-Att^{sub} Patterns. MSD query *msd_fin* is *V;ACT;PRS;POS;IMP;3;PL*. Number of examples (*No of*) and accuracy (*Acc*) are shown per selection with query and per group pattern. For each query, we list all extracted lemma patterns (sorted by frequency in a decreasing order) along with one segmented lemma example (in parentheses) mapped to the pattern.